Reasoning and querying bounds on differences with layered preferences

Reasoning and Querying Bounds on Differences with Layered Preferences

Luca Anselma[1], Alessandro Mazzei[1], Luca Piovesan[2] and Paolo Terenziani[2]

[1] Dipartimento di Informatica, Università di Torino, corso Svizzera 185, 10149 Torino, Italy
[2] DISIT, Università del Piemonte Orientale "A. Avogadro", Alessandria, Italy

{anselma,mazzei}@di.unito.it, {luca.piovesan,paolo.terenziani}@uniupo.it

Abstract— Bounds on differences are widely used in AI to model binary constraints regarding different dimensions, such as time, space, costs, calories, etc. Representing and reasoning with them is an important task in several areas such as knowledge representation, scheduling and planning. Researchers are increasingly focusing on the treatment of fuzzy or probabilistic constraints, to deal with preferences and/or uncertainty. Current approaches to constraints with preferences focus on the evaluation of the optimal (i.e., with highest preference) solutions for the set of constraints and propose a wide range of alternative operators to combine preferences within constraint propagation. However, in decision support tasks, finding a specific (though optimal) solution is not the main goal, but rather it is more important to identify the "space of solutions" (i.e., the *minimal network*) with their preferences, and to provide users with query answering mechanisms to explore it. We propose the first approach that addresses such a need by (i) supporting *user-defined* layered scales of preferences (e.g., Low, Medium, High, Very High), (ii) proposing a family of extensions of bounds on differences constraints to deal with such *layered preferences*, (iii) defining a family of *reasoning algorithms* to evaluate the minimal network, which is *parametric* with respect to the basic operations to combine preferences (and the scale of preferences), and (iv) providing suitable *query-answering facilities*. The properties of the family of approaches are also analyzed.

Keywords — Temporal constraints with preferences; Temporal reasoning; Temporal constraint propagation

1 INTRODUCTION

The treatment of fuzzy information in general, and *preferences* in particular, plays an important role in intelligent systems, and, specifically, in Decision Support [74]. As a result, many "classical" AI frameworks have been and are being extended to consider preferences. In this paper, we focus on one of them, Bounds on Differences (BoDs), which are an important tractable subclass of the general class of Constraint satisfaction problems (CSPs).

CSPs are the subject of an important stream of research in Artificial Intelligence (AI), since they provide a common basis to represent, analyze and solve many seemingly unrelated families of problems. Coping with CSPs is, in general, an NP-hard problem. However, polynomial tractable problems can be obtained by restricting the class of constraints. A tractable class of constraints is the one of BoDs [16], a particular class of linear inequality problems which consist of a conjunction of constraints $c \leq x - y \leq d$, where the difference between variables x and y is bounded between the values c and d (the domain can be either discrete or dense). BoDs assume a particular importance because many real-world problems, especially spatial and temporal problems, can be easily modelled as sets of BoDs. BoDs may have different interpretations. For instance, x and y may be interpreted as time points, c and d are respectively their minimum and maximum distance on the timeline (this is the basis of the Simple Temporal Problem framework – STP,

see [19] – which is a milestone in AI temporal reasoning). Alternatively, c and d can be the spatial distance between points in space (see, e.g., [47, 59]), or calories [3].

For the sake of brevity, in the rest of the paper we propose examples relying on the temporal interpretation of BoDs, but, very importantly, all the possible BoD interpretations are supported by our methodology.

A number of approaches in AI provide ***reasoning*** algorithms that propagate a set of BoDs repeatedly applying two operators: the *resume* $\oplus$ and the *extension* $\odot$ operators. Given two constraints concerning the same pair of variables *i* and *j*, the resume operator $\oplus$ gives as output the new constraint "merging" them; given a constraint between two variables *i* and *j* and a constraint between *j* and *k*, the *extension* operator $\odot$ is used to determine the new "implied" constraint between *i* and *k*. Reasoning can be performed to achieve different tasks: to check the *consistency* of the constraints or to find a *solution* or *scenario* (i.e., an instantiation of all the variables that satisfies all constraints), or to compute the *minimal network*. A minimal network is a set of constraints equivalent to the original ones (i.e., with the same solutions) such that the minimum and maximum implied distances between each pair of variables are made explicit (see [19] for a formal definition).

Example 1. To simplify the discussion, we consider an easy STP problem. Let t_1, t_2 and t_3 be time points, let granularity be minutes, and let KB be the following set of BoD constraints: KB={$10\leq t_2-t_1\leq 15$, $20\leq t_3-t_2\leq 30$, $25\leq t_3-t_1\leq 40$}, i.e., t_2 is between 10 and 15 minutes after t_1, t_3 is between 20 and 30 minutes after t_2, and t_3 is between 25 and 40 minutes after t_1. As an example of extension and resume, the *extension* operation $10\leq t_2-t_1\leq 15 \odot 20\leq t_3-t_2\leq 30$ gives as result the BoD $30\leq t_3-t_1\leq 45$ (intuitively, if the difference between t_2 and t_1 is in the range [10,15] *and* the difference between t_3 and t_2 is in the range [20,30], the implied difference between t_3 and t_1 is in the range [30,45]); the *resume* operation $30\leq t_3-t_1\leq 45 \oplus 25\leq t_3-t_1\leq 40$ gives as result the BoD $30\leq t_3-t_1\leq 40$ (intuitively, if the difference between t_3 and t_1 is both in the ranges [30,45] and [25,40], the implied difference is in the range [30,40] – see the definition of the operators in Section 3). KB is consistent, and {$t_1=0$, $t_2=10$, $t_3=30$} is a scenario (solution) of KB. The *tightest* constraints implied by KB, that is the minimal network, are KB'={$10\leq t_2-t_1\leq 15$, $20\leq t_3-t_2\leq 30$, $30\leq t_3-t_1\leq 40$}. In particular, notice that in the minimal network we inferred that the minimum distance between t_1 and t_3 is 30. ■

While in several tasks (e.g., in scheduling) finding a *solution* is sufficient, in other tasks it is necessary to determine a compact representation of *all* the possible solutions in the form of a *minimal network* of the constraints. This is the case, e.g., when considering *decision support* systems, and/or when supporting users in *mixed-initiative* approaches. In such cases, providing users with a specific solution would be restrictive and not user friendly, since the final choice of a specific solution has to be left to the users.

In such contexts, also ***query answering*** is important, to give users a way to explore the space of solutions.

Example 2. Given KB, the user might ask:
(Q1) *May I execute t_3 28 minutes after t_1?*
(Q2) *If I performed t_1 at 21, when (i.e., in which range of time) can I perform t_2 and t_3?* ■

The literature shows that the tasks of reasoning and of query answering are strictly connected: in fact, the *correctness* of *query answering* can be granted only if the *minimal network* with the tightest constraints is provided by the reasoning task. Indeed, computing the *minimal network* constraints is a fundamental task, and many efforts have been devoted to it [19, 56, 61, 69, 71].

Example 3. Let us suppose that a not complete reasoning process is employed to a knowledge base KB, so that not all the tightest constraints are obtained (or, in technical terms, the reasoning process does not compute the *minimal network*). E.g., let us suppose that KB"=$\{10\leq t_2\text{-}t_1\leq 15,\ 20\leq t_3\text{-}t_2\leq 30,\ 27\leq t_3\text{-}t_1\leq 40\}$ is obtained. Considering KB", user's query Q1 in Example 2 receives an *erroneous* positive answer. Thus, the unavailability of the minimal network (with the *tightest* constraints) cannot guarantee that the answers are *correct* and, thus, *reliable*. ■

The constraints that we introduced up to this point are "*crisp*" constraints, i.e., the values of the distances between variables are all "*equally possible/preferred*". Approaches based on crisp constraints are based on the general framework of classical CSP and inherit from it some important limitations in flexibility and support for uncertainty; in fact, while classical CPSs' constraints are hard, in real-world problems often constraints are not hard, and a **preference** among the feasible solutions has to be accounted for [21].

An important example can be found in the execution of clinical treatments with temporal constraints. Usually, in the medical context it is not possible (and often not useful) to express crisp temporal constraints. On the contrary, constraints are typically recommendations, to be respected as much as possible, depending on the specific cases.

Example 4. In the case of antibiotic drugs, drug administrations must be performed with fixed delays (temporal distances) between them. For instance, the antibiotic nalidixic acid is a drug used for the treatment of urinary tract infections. It must be administered twice a day with a delay of 12 hours between two consecutive administrations. This is not a strict recommendation: it has a "high" preference and the delay between two administrations can also be of 11-13 hours (with a "medium" preference) or of 10-15 hours (with a "low" preference). In case a patient suffers from urinary tract infection, a physician may want to know when (in which ranges of time) nalidixic acid can be administered, and what is the preference of the different possibilities. Notably, real-world scenarios can be more complex and can involve constraints deriving from different sources. Let us consider a case of a comorbid patient treated both for urinary tract infection and for gastroesophageal reflux with a calcium carbonate administration after meals, when needed. The concomitant administration of the drugs can lessen the effect of the nalidixic acid. For such a reason, nalidixic acid must be administered at least three (with "low" preference) or four (with "high" preference) hours after calcium carbonate. Moreover, when executing clinical treatments, also patient preferences must be considered. In our example, we hypothesize that a patient takes calcium carbonate after breakfast and dinner, and she has the following preferences regarding meal times: breakfast (7am with "high" preference, 6am-7am with "medium" preference, 6am-8am with "low" preference), dinner (6pm-8pm with "high" preference, 8pm-9pm with "medium" preference, 5pm-10pm with "low" preference). Each breakfast can last up to one hour, while dinner can last up to two hours. ■

To deal with issues such as the ones discussed above, a vast stream of research provided a fuzzy extension to the CSP formalism, by replacing classical *crisp* constraints with soft *non-crisp* constraints modeled by fuzzy relations. A number of approaches based on the Fuzzy Constraint Satisfaction Problem [21] have been devised (see related work in Section 2), and a lot of attention has been devoted, specifically, to non-crisp extensions of BoD constraints (see Section 2.3). In this work, we focus on BoDs with preferences, and we propose two main types of contributions.

First, despite the variety of the CSP approaches in the literature, the approaches about *preferences* in the area have mostly focused their attention on the problem of determining the optimal (i.e., the most preferred) *solutions*. On the other hand, for many applications (e.g., in decision support tasks), providing users with specific solutions may be restrictive. Instead*, as widely considered in the case on crisp constraints*, users may want to be provided with a

(compact representation of) the "space of solutions" (minimal network), and with facilities to explore such a space, in order to be able to choose among possible solutions. This is the case, for instance, in our long-term project GLARE (Guideline Acquisition, Representation and Execution) [63], when supporting physicians in the application of clinical guidelines to patients. Notice that it would be very restrictive and definitely not user-friendly to impose a specific time of execution of the actions to the physician, even if it is the optimal one (see, e.g., [5]). We overcome such a general limitation by proposing an approach coping with BoDs with preferences that aims at (i) evaluating the *minimal network of the constraints with preferences*, and at (ii) proposing query-answering facilities to explore it[1]. In such a way, our work makes available the advantages of non-crisp CSP approaches also for decision-support tasks.

Second, CSP approaches in the literature propose quite different solutions to the crucial problems of (i) how to define preferences, (ii) how to associate preferences with constraints and (iii) how to combine the preferences in the constraints while applying the *resume* ($\oplus$) and *extension* ($\odot$) operators discussed above. The variety of solutions indicates that there is not an overall best solution to (i)–(iii): there are several alternative possibilities, and the choice between them is usually task- and/or domain-dependent. As a consequence, the goal of our approach is to provide a general framework, covering as much as possible all the different possibilities. Specifically, we propose a general and homogeneous framework that advances the state of the art of non-crisp BoD constraints in five main directions.

(1) We support the treatment of non-numeric, finite and totally ordered scales of preferences (e.g., <Low, Medium, High, Very High>) called ***layered*** preferences henceforth. Since the definition of a scale of preferences is usually context- and\or task-dependent, our approach is *parametric* with respect to any finite scale of preferences.

(2) We consider two different ways (formalisms) to associate layered preferences with BoD constraints: BoD constraints with preferences (see Section 3) and BoD constraints with Pyramid preferences (see Section 4).

(3) For both formalisms, we support *user-defined* definitions of the basic operations used to compose the preferences in the constraints while performing the *resume* ($\oplus$) and *extension* ($\odot$) operations for propagating the constraints.

(4) We propose a *family* of constraint propagation algorithms to compute the minimal network of BoD constraints with preferences, covering the different possibilities provided by points (2) and (3) above, and we study their properties.

(5) We provide a *query-answering* mechanism to support the user analysis of the minimal network with preferences.

The paper is structured in the following way. In Section 2, we discuss related works. In Section 3, we introduce our basic representation of BoD constraints with layered preferences, and provide a general constraint propagation algorithm grounded on the well-known path-consistency algorithm [44, 66], which computes the minimal network of BoDs and propagates the layered preferences (in any finite input scale) on the basis of *user-defined* composition operations; in other words, our algorithm is *parametric* with respect to the composition operations provided by users. In Section 4, we specialize our general approach to the case in which preferences can be represented by "pyramids" (which is relevant in many real-world contexts) and specialize our constraint propagation algorithm to such a case. In Section 5, we further specialize our approach by showing that, in case the operations which compose constraints

[1] To the best of our knowledge, the approach by Terenziani et al. [60] is only other one in the literature about BoDs addressing such issues; the advances with respect of such an approach are widely discussed in Section 7 of this paper.

constitute a closed semiring, a more efficient constraint propagation algorithm can be used. To substantiate such a general claim, we consider a specific instance of the basic operations and we propose an algorithm that computes the *minimal network* in polynomial-time. Finally, in Section 6, we propose a query language and a query-answering approach to support users in the analysis of the minimal network, and in Section 7 we propose comparisons and conclusions.

2 RELATED WORK

BoDs constraints have been widely studied in AI since they constitute a simple and nevertheless useful version of CSP. In classical CSPs [41, 44] problems are represented in terms of a set of variables and a set of constraints over such variables. CSPs are a very powerful formalism, and several real-life problems (e.g., assignment and scheduling problems, network management, transport problems, VLSI design, robotics problems, etc.) can be represented and solved through them. However, CSPs have also evident limitations. Indeed, in classic CSPs, constraints can only be *crisp*: in any (complete) assignment of values to the variables, each constraint is either satisfied or not and only those assignments that satisfy all the constraints are considered *solutions* of the problem. Thus, classic CSPs cannot naturally cope with scenarios where knowledge is neither complete nor crisp. For such a reason, in the last thirty years, a lot of work has been devoted to cope with incomplete or non-crisp constraint problems. Several classes of constraints have been considered, and different types of preferences. One of the most prolific fields has been the one considering CSPs and fuzzy preferences. For such a reason, in Section 2.1 we present approaches coping with non-crisp CSPs and CSPs with preferences associated with constraints. Then, since one of the main goals of our approach is to provide a parametric approach in which the management of preferences (i.e., the function to combine them) is given as an input, in Section 2.2 we go in depth in the description of the different strategies to combine preferences. In Section 2.3, we describe the class of BoD constraints and the approaches in literature coping with non-crisp BoDs. Finally, in Section 2.4 we discuss other approaches coping with non-crisp temporal problems that are not based on BoDs.

2.1 Non-crisp CSPs and CSPs with preferences

Several forms of non-crisp CSPs have been proposed in the last thirty years. For instance, *probabilistic CSPs* [27] allow to cope with scenarios in which the problems can be not completely known a priori (e.g., whether a specific constraint belongs to a problem). To do that, probabilistic CSPs allow to represent and to cope with the fact that a specific constraint has a probability p to be part of the problem. In such problems, the goal is to find a complete assignment to the variables that has the highest probability to be a solution of the problem. On the other hand, *weighted CSPs* [57] have been developed to cope with over-constrained CSPs. They associate costs or weights with the constraints in the problem. For such problems, the goal is no more to find a solution that satisfies all the constraints, but rather to find a solution that maximizes the sum of the weights of the satisfied constraints. *Partial CSPs* [30] are a particular case of weighted CSPs in which all the constraint weights are equal to 1.

One of the most common situations when coping with real-world problems is the one in which not all the constraints, or the values in the domains of the variables, have the same importance. In such situations is quite natural to associate some levels of *preference* with the constraints or with the values allowed for the variables. In such contexts, preferences allow to express “less strict aspects of a problem, such as desires, satisfaction levels, rejection degrees and

costs" [50]. The general idea underlining the introduction of preferences into CSPs is that a constraint is no longer an element with a binary satisfiability (i.e., satisfied or not satisfied), but an element with several levels of satisfiability.

Starting from the early 1990s, a wide area of research has been devoted to the definition of formalisms and to the development of techniques to solve CSPs with preferences. We can distinguish among at least three kinds of them depending on the type of preferences: *quantitative* preferences, *qualitative* preferences and *bipolar* preferences.

Among the approaches coping with quantitative preferences, the *fuzzy CSPs* [12, 13, 22, 49, 51, 54] associate variable assignments with preferences belonging to the interval [0,1]. *Semiring-Based* (fuzzy) *CSPs* [12, 13] have a particular importance for the work in this paper, since they adopt *semirings* to cope with fuzzy CSPs and propose forms of local consistency to solve them. Similarly to Semiring-Based CSPs, *Valued CSPs* [55] rely on a different structure, *monoids*, which are composed by an ordered set of elements (e.g., preferences) and an operation to aggregate two elements. The main difference between Semiring-Based and Valued CSPs is that the latter cannot cope with partially ordered sets of preferences. This choice limits their use, for instance, when coping with preferences that are the result of the combination of several criteria through Pareto-like approaches that lead to a partial order.

As regards qualitative preference approaches, *CP-nets* [15] model preferences in a qualitative and conditional way (e.g., if the main course is meat, red wine is better than white wine).

On the other hand, *bipolar* preference approaches [11, 23, 24] cope with the fact that preferences can be either positive or negative, and these two kinds of preferences are one the opposite of the other.

2.2 Combining preferences in fuzzy CSPs

When dealing with constraints with preferences, there is the need of defining a criterion to combine them in order to evaluate the preference of a solution based on the preferences of the single constraints. In fuzzy CSPs, solutions are usually evaluated considering a pessimistic policy (also called *weakest-link optimality* policy): the preference of a solution is the minimum of the preferences of its constraints. However, different domains might require different policies. For example, Grabisch et al. [32] define several types of functions for combining fuzzy elements and discuss their properties. They distinguish among three main classes of functions: *conjunctive*, *disjunctive* and *compensative* ones. *Conjunctive functions* connect elements through operators that are similar to the logical "and". Thus, using conjunctive functions, the global preference is high if and only if all the preferences of the constraints are high. An example of conjunctive class of operators is the Triangular Norm[2] (t-norm) [29, 38, 42]. The literature proposes different types of t-norms. Among them, for instance, the Gödel (or Minimum) t-norm combines elements through the *min* function. Thus, the weakest-link optimality policy can be considered a Gödel t-norm aggregation. Another relevant form of t-norm is the Product t-norm, which combines preferences with the arithmetic product operator. On the other hand, *disjunctive functions*, the second class of functions, connect elements through operators similar to the logical "or". Thus, the global preference is low if and only if all the partial preferences are low. T-conorms are examples of disjunctive functions, and they are dual to t-norms. For instance, the Maximum t-conorm combines elements through the *max* function. Finally, in *compensative functions* low-preference elements compensate high-preference elements. Mean operators (e.g., arithmetic, geometric, harmonic, root power means) and median operators are examples of compensative functions. Also the *utilitarian* policy, presented in [64], can be seen as a kind of compensative function.

[2] Notice that t-norms and t-conorms are binary operators. However, due to their associative property, they can be applied to sets of elements.

It is worth stressing that the criterion defined by aggregation functions for fuzzy elements can, in many cases, also cope with ordered scales of preferences (e.g., the Gödel t-norm) or they can be generalized to do that. For instance, the Drastic t-norm, defined as $T_D(a,b) = \begin{cases} b & if\ a = 1 \\ a & if\ b = 1 \\ 0 & otherwise \end{cases}$ can be generalized to cope with two preferences a and b belonging to an ordered scale of preferences, given a top $\top$ and a bottom $\bot$ elements in the scale, as $T_D(a,b) = \begin{cases} b & if\ a \equiv \top \\ a & if\ b \equiv \top \\ \bot & otherwise \end{cases}$.

2.3 Non-crisp BoDs and BoDs with preferences

BoDs [16] constitute a subclass of CSPs that received much attention due to the fact that many real-world problems, especially spatial and temporal problems, can be easily modelled as sets of BoDs, and several problems (i.e., consistency checking) can be managed in polynomial time with BoDs. For instance, Simple Temporal Problems (STPs) [19] represent temporal problems through sets of BoDs, where variables represent time points and constraints limit the span of time between each pair of time points. On the other hand, in several spatial problems (e.g., the minimal routing problem) variables represent spatial points and the constraints limit the distances between them. In contrast with CSPs, BoD problems can be solved in polynomial time. When coping with BoDs, however, finding one or more solutions of the problem is not always the final goal. Rather, finding the minimal network[3] is in many cases more important. Indeed, problems like STP or the all-pairs shortest paths problem [28, 58] are managed through algorithms that find the minimal network. Besides supporting the identification of specific solutions of a problem, the minimal network allows to efficiently answer queries considering the set of the solutions of a problem, like "Can event A temporally precede event B?" or "Does event A temporally precede event B in all the solutions?". Without the minimal network, answering such types of queries could require computing all the solutions of a problem. Luckily, in BoDs both computing the minimal network of constraints of a problem and answering queries using it can be done in polynomial time and space (e.g., using algorithms like the Floyd-Warshall's one [28]).

BoDs[4], as well as classic CSPs, have been extended to cope with non-crisp aspects. For instance, Andolina and Terenziani [62] have considered BoD constraints associating a probability value with each possible distance (only discrete domains for distances have been considered). They used Floyd-Warshall's algorithm to propagate such temporal constraints and to evaluate the minimal network of distances, adopting new operators to combine constraints taking into account both distances and probabilities. Considering preferences, Andolina et al. [60] extended STP to associate numeric preferences with distances defined over discrete domains. They used Floyd-Warshall's algorithm to compute the minimal network of distances by propagating such temporal constraints, adopting new "combination" operators that consider both distances and numeric preferences. The theory of c-semiring is used to grant that the tightest constraints (i.e., the minimal network) are computed between each pair of variables. Query answering on the resulting minimal network is also supported. The recent approach by Anselma et al. [4] generalizes and extends the

[3] Notice that structures similar to the minimal network can be obtained also for other classes of CSPs. However, computing them often requires non-polynomial space and/or time.

[4] Actually, there are also other approaches coping with the more general class of linear inequalities. For instance, the work in [72] proposes an approach coping with fuzzy linear inequalities. However, in this section we only focus on BoDs.

approach in [60] by coping with continuous domains and by supporting user-defined layered preferences. It is worth stressing that layered preferences, or "pyramid" ones (see Section 4) can represent a wide range of phenomena. Indeed, they can be seen as a discretization of Gaussian functions, which are widely used in preference representation. Khatib *et al*. [37] extended constraint-based temporal reasoning (specifically, the STP and the TCSP frameworks) for reasoning about temporal preferences, and they examined the complexity of the resulting formalism. They showed that, by exploiting c-semirings in the treatment of preferences, it is possible to achieve tractability. The approaches by Andolina et al. [60], Anselma et al. [4] and Khatib *et al*. [37] are the closest ones to ours: a detailed comparison with such approaches is reported in Section 7 after the description of our approach.

2.4 Other non-crisp temporal approaches

To conclude this section, we briefly mention other approaches to non-crisp constraints, specifically devoted to the treatment of temporal phenomena. Bugarín *et al*. [17] discussed a wide number of approaches where time is included as an additional decision variable in fuzzy propositions and rules. Considering only CSP approaches, a number of temporal reasoning approaches based on fuzzy CSPs [6] have been devised. For instance, Barro *et al*. [8] introduced a model for representing and handling fuzzy temporal preferences. They exploited the formalism of possibility theory for defining the concepts of dates, time extents, and intervals; the relations between them are interpreted as constraints on the distances and projected to fuzzy temporal constraint satisfaction networks. Vila and Godo [70] based a propositional temporal language on fuzzy temporal constraints to deal with domains where the knowledge is of propositional nature and it is required to explicit handle time, imprecision and uncertainty. The formalism relies on natural possibilistic semantics for dealing with the uncertainty arising by the fuzziness of temporal constraints. They also presented an inference system based on specific rules dealing with the temporal constraints and a general fuzzy modus ponens rule and they showed that the reasoning is sound. Kamide and Koizumi [35] have recently proposed an inconsistency-tolerant probabilistic tree logic. Recently, Gammoudy *et al*. [31] have proposed to model Allen's qualitative relations between fuzzy time intervals, while Billet *et al*. [10] have considered "ill-known" time intervals.

In the area of scheduling/planning, many approaches have taken into account the distinction between controllable and non-controllable temporal constraints. The simple temporal network with uncertainty (STNU) [33] has been proposed to extend STP introducing set-bounded uncertainty to deal with events that are non-controllable. [65] also extended STNU providing a probabilistic representation of the uncertainty. In this extension, information regarding the distribution of non-controllable events enables planning for more likely outcomes. Many works considered temporal planning with uncertainty, for example simple temporal problem (STP) under uncertainty [34], conditional STNU [33], disjunctive temporal problems with uncertainty [67], probabilistic temporal plan networks [53], and temporal plan networks with uncertainty [25]. Fang *et al*. [26] introduced pSTN (probabilistic simple temporal network), a probabilistic approach for representing temporal problems with bounded risk along with a utility over event timing. To grant robust scheduling, the authors also introduced a constrained optimisation algorithm that achieved efficiency and compactness for strong controllability [68]. Yorke-Smith *et al*. [75] proposed an integrated framework supporting both uncertainty and preferences related to the problem of controllability. Recent planning approaches have focused on the treatment of temporal constraints with preferences [39, 46].

A number of approaches dealt specifically with the representation of "non-crisp" temporal constraints between points and/or intervals, and with reasoning on them by means of constraint propagation [43]. In this main stream of research, a basic distinction is made between qualitative and quantitative temporal constraints (see, e.g., [69]). Qualitative

constraints (e.g., "A during B") are non-metric constraints modelling the relative position of actions/events, while quantitative constraints are metric constraints on events/facts (e.g., "A started on 1/1/2018 at 12:00", "the delay between A and B is 20-25 minutes"). Concerning qualitative constraints, i.e., non-metric constraints, Ryabov et al. [52] augmented the Allen's basic interval relations with probabilities and uncertain relations are represented as disjunctions of Allen's probabilistic basic relations. The authors defined the operations of inversion, composition, and addition for probabilistic interval relations and used a path-consistency algorithm for propagating such constraints. More recently Mouhob and Liu [45] proposed a similar probabilistic approach, inscribed in the general probabilistic CSP framework. Another approach in the same line of research, is the one of Badaloni and Giacomin [6], who devised an extension of Allen's interval-based framework by associating a preference degree with relations between intervals.

3 A GENERALIZED AND PARAMETRIC APPROACH TO BoD WITH PREFERENCES

In this section, we provide a general approach extending BoDs with preferences. For the sake of generality, our approach applies to any layered scale of preferences (see Definition 3 below), and to the operations to "combine" preferences (see explanations below). Thus, users can simply give in input their own scale and combination operations to customize our approach to the specific needs of their applications.

3.1 BoD with preferences

BoDs are linear inequalities of the form $c \leq x - y \leq d$, with $c, d \in \mathbb{D}$, and x and y are variables on the same domain, and $\mathbb{D}$ is a subset of $\mathbb{R}$ closed under addition[5]. A BoD $c \leq x - y \leq d$ can be seen and represented as the interval [c,d] denoting all the possible values for the difference $x - y$. Different interpretations of BoDs have been used in different domains. For instance, in temporal reasoning, variables may represent events and [c,d] an interval (range) of temporal distances; in spatial applications, variables may represent points in space and [c,d] an interval (range) of spatial distances. For the sake of compactness, we explicitly introduce the notion of interval of differences that we term "admissibility interval" (for a difference).

Definition 1. Admissibility Interval (AdI). Given two values $c, d \in \mathbb{D}$, we denote by [c,d] the interval (termed *admissibility interval*) containing all the possible values $v \in \mathbb{D}, c \leq v \leq d$. ■

When $c > d$, the interval is degenerate.

We denote by $\mathbb{A}^{\mathbb{D}}$ the domain of admissibility intervals over $\mathbb{D}$. For generality, in the following we denote by $\mathbb{V}$ the domain of variables. We can thus represent BoDs as follows:

Definition 2. Bound on Difference (BoD). In our representation, a BoD over a domain $\mathbb{D}$ is a constraint $\langle x, y, [c, d] \rangle$ of the form $\mathbb{V} \times \mathbb{V} \times \mathbb{A}^{\mathbb{D}}$ associating an admissibility interval with an ordered pair of variables[6]. ■

In our approach, we support the possibility of associating layered preferences with BoDs, and, in particular, with the (values in the) admissibility intervals. Layered preferences are usually domain- and/or task-independent. We simply impose that they form a finite and totally ordered set.

[5] Actually, $\mathbb{D}$ could be a more abstract algebraic structure, such as an ordered semigroup. However, for the sake of simplicity of the proofs in the paper, here and in the following we consider a generic subset of $\mathbb{R}$ closed under addition.

[6] The variables can represent for example time points or spatial points.

Definition 3. Scale of Layered Preferences (SLP). An SLP (or "scale", for short) S_r of ***cardinality*** *r* consists in an enumerative set $\{p_1, \ldots, p_r\}$ of r labels (with r>0), and a strict total ordering relation < over the set. For easiness of notation, we denote an SLP by an ordered list $\langle p_1, \ldots, p_r \rangle$, such that $\forall i, 1 \leq i < r, p_i < p_{i+1}$. ■

Terminology. Given an SLP S_r of ***cardinality*** *r,* we indicate by $S_r(i)$ the i[th] value in the scale S_r $(1 \leq i \leq r)$. ■

The domain of SLPs is denoted as $\mathbb{S}$.

For instance, an SLP in $\mathbb{S}$ dealing with Example 4 is S_3^{ex}:<low, medium, high>, and $S_3^{ex}(2)$ is "medium".

We extend the "standard" BoD approach by introducing preference functions, as a tool to associate preferences with difference values.

Definition 4. Preference function. A preference function $Pref_{S,D}$ over the domain $D \in \mathbb{A}^{\mathbb{D}}$ and with scale $S \in \mathbb{S}$ is a total function $Pref_{S,D}$: D→S. ■

Let $\mathbb{P}^{\mathbb{D}}$ denote the domain of preference functions, defined as above. A BoD constraint with preferences associates a preference with each possible value of difference between two variables.

Definition 5. BoD with preferences (P_BoD). Given a scale $S \in \mathbb{S}$, a P_BoD is a constraint $\langle x, y, [c, d], Pref_{S,[c,d]} \rangle$ of the form $\mathbb{V} \times \mathbb{V} \times \mathbb{A}^{\mathbb{D}} \times \mathbb{P}^{\mathbb{D}}$ associating a preference with each possible difference value between an ordered pair of variables. ■

Example 4(a). In Figure 1, for instance, <NA1,NA2,[10,15],{(10,low),(11,medium),(12,high),(13,medium),(14,low),(15,low)}> is a P_BoD representing the constraint between the two successive administrations of nalidixic acid NA1 and NA2 of Example 4. Notice that, in this example, the preference function Pref is given in the form of a set of pairs (d, s), with $d \in D$ and $s \in S$, and $S = \langle low, medium, high \rangle$. The formalization of the complete example is given (following the PyP_BoD_G formalism) in Figure 3 in Section 4.1.

It is well known in the literature that a set of BoD constraints can be represented as a graph and that the graphical representation can be exploited by reasoning algorithms. We define a graph of P_BoD constraints as follows.

Definition 6. Graph of BoD with Preferences (P_BoD_G). Given an SLP S of cardinality r, a set B of P_BoD $\langle x, y, [c, d], Pref_{S,[c,d]} \rangle$ can be represented as a graph G=<V,E> and a labelling function λ, where the set of nodes V is the set of variables in B, the set of edges E ⊆ V×V and the labelling function λ: E → $\mathbb{A}^{\mathbb{D}} \times \mathbb{P}^{\mathbb{D}}$. ■

Example 4(b). In Figure 1, for instance, we show an edge of the P_BoD_G graph (i.e., a single P_BoD) representing the constraint between the two successive administrations of nalidixic acid NA1 and NA2 of Example 4.

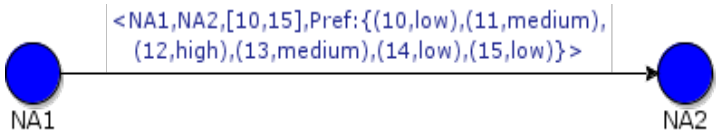

Figure 1. A P_BoD representing the constraint between the two successive administrations of nalidixic acid NA1 and NA2 of Example 4.

3.2 Propagation of binary constraints

In many constraint-based approaches, local consistency algorithms are employed for constraint propagation [44, 66]. An important form of local-consistency-enforcing algorithm are path consistency algorithms [44, 66]. We report in Figure 2 a general path-consistency algorithm that uses the general operations of resume $\oplus$ and extend $\odot$ between constraints. In this section, we do not dwell on the specialized and more efficient forms of path consistency algorithms for constraint satisfaction problems [9, 20] and for spatio-temporal constraints [40, 73].

The path-consistency algorithm accepts as parameters a labelling function $\lambda: E \rightarrow \mathbb{A}^{\mathbb{D}} \times \mathbb{P}^{\mathbb{D}}$ defined on the edges of the graph, the vertices V, the edges E, an "*extension*" operator $\odot$, a "*resume*" operator $\oplus$, as well as two additional parameters, i.e., the *identity* for $\odot$ (indicated as $\mathbf{1}$), and the *identity* for $\oplus$ (indicated as $\mathbf{0}$). We assume that $\lambda(i,j)=\mathbf{0}$ if $(i,j)\notin E$ and that $\lambda(i,i)=\mathbf{1}$. Notably, the path-consistency algorithm is parametric with respect to the $\oplus$ and $\odot$ operations. Indicating as 1,2,…,n the vertices in V, with n=|V|, the resume operator $\oplus$ takes as input two constraints concerning the same pair of variables *i* and *j*, and gives as output the new constraint "merging" them; the *extension* operator $\odot$ takes as input a constraint between two variables *i* and *j* and a constraint between *j* and *k*, and determines the new "implied" constraint between *i* and *k*.

The path-consistency algorithm revises each constraint L_{ij} between the nodes i and j considering each pair of constraints L_{ik} and L_{kj} and applying the operations of resume $\oplus$ and extend $\odot$ between them. This is iterated until a quiescence state is reached or until some constraint becomes empty (in such a case the network is inconsistent).

Notice that, in general, the algorithm is not guaranteed to terminate, since, because of the generality of the resume $\oplus$ and extend $\odot$ operators, there is no guarantee that at each iteration of the main loop the network gets tighter, so that a state of quiescence, in which no constraints can be further changed, can be eventually reached.

Path-Consistency(λ,V,E,⊕,⊙,**0,1) algorithm**

```
1. n ← |V|
2. for i←1 to n do
3.    for j←1 to n do
4.          Lij ←λ(i,j)
6. repeat
7.     for k←1 to n do
8.           for i←1 to n do
9.               for j←1 to n do
10.                 Lij ← Lij ⊕ (Lik ⊙ Lkj)
11.until no constraint is changed
12.return L
```

Figure 2. Path-Consistency algorithm.

As an example of $\oplus$ and of $\odot$, in the case of crisp BoD constraints, $\oplus^{B}$ and of $\odot^{B}$ are defined as follows. The resume operator $\oplus^{B}$ computes the intersection of the admissibility intervals. If the result is a degenerate interval $[c, d]$ with $c > d$ then an inconsistency has been detected.

Definition 7. Resume ($\oplus^{B}$). Given two BoDs $C' = \langle x, y, [c', d'] \rangle$, and $C' = \langle x, y, [c'', d''] \rangle$, $C' \oplus^{B} C'' = \langle x, y, [\max(c', c''), \min(d', d'')] \rangle$. ■

Definition 8. Extend ($\odot^{B}$). Given two BoDs $C' = \langle x, y, [c', d'] \rangle$, and $C' = \langle y, z, [c'', d''] \rangle$, $C' \odot^{B} C'' = \langle x, z, [c' + c'', d' + d''] \rangle$. ■

As an example of extension and resume, $= \langle t_1, t_2, [10,15] \rangle \odot^{B} \langle t_2, t_3, [20,30] \rangle = \langle t_1, t_3, [30,45] \rangle$ and $\langle t_1, t_3, [30,45] \rangle \oplus^{B} \langle t_1, t_3, [25,40] \rangle = \langle t_1, t_3, [30,40] \rangle$.

Notably, it is well known that, for crisp BoD constraints, the outer cycle of the algorithm enforcing path-consistency is not necessary. The minimal network of a set of BoD constraints can be computed directly by the three nested loops, and, specifically, by Floyd-Warshall's algorithm and its optimized versions [19, 48].

3.3 Propagation of P_BoDs

We adopt the standard path-consistency algorithm in Figure 2 to propagate our P_BoD constraints, by providing an appropriate definition of the $\oplus$ and $\odot$ operators, that we indicate by $\oplus^{PB}$ and $\odot^{PB}$. Indeed, $\oplus^{PB}$ and $\odot^{PB}$ operate separately on the two components of P_BoD constraints. As regards the admissibility intervals (i.e., the minimal distance *c* and the maximal distance *d*) in the P_BoD constraints, we operate in the standard way, using the $\oplus^{B}$and $\odot^{B}$ operators in Definitions 7 and 8. However, we have also to indicate how to resume ($\oplus^{\text{pref}}$ operator) and extend ($\odot^{pref}$ operator) the preferences associated with distances in the constraints.

As discussed in Section 2, several different proposals have been already presented in the literature for combining preferences, and the choice among them seems to be closely dependent on the specific task and/or goal of the approaches. Indeed, we aim at providing the maximum possible generality, so that we also envision the possibility that new approaches propose new combination functions. For such a reason, we propose a general and flexible approach, which is parametric with respect to the operations $\oplus^{pref}$ and $\odot^{pref}$ to combine preferences (in other words, our approach takes the $\oplus^{pref}$ and $\odot^{pref}$ operations as parameters provided by users).

Our general definition of $\oplus^{PB}$ and $\odot^{PB}$ is provided in the following.

The resume operator $\oplus^{PB}$ computes the intersection of the admissibility intervals (as the $\oplus^{B}$ operator); if the result is a degenerate interval $[c, d]$ with $c > d$ then an inconsistency has been detected. Moreover, for each value *v* in the resulting admissibility interval $[\max(c', c''), \min(d', d'')]$, $\oplus^{PB}$ combines through the $\oplus^{pref}$ operator the preference of *v* in the first P_BoD constraint (i.e., ${Pref'}_{S,[c',d']}(v)$) and in the second P_BoD constraint (i.e., ${Pref''}_{S,[c'',d'']}(v)$).

Definition 9. Resume ($\oplus^{PB}$). Given two P_BoDs $C' = \langle x, y, [c', d'], {Pref'}_{S,[c',d']} \rangle$ and $C'' = \langle x, y, [c'', d''], {Pref''}_{S,[c'',d'']} \rangle$, $C' \oplus^{PB} C'' = \langle x, y, [\max(c', c''), \min(d', d'')], {Pref}^{\oplus}{}_{S,[\max(c',c''),\min(d',d'')]} \rangle$, with ${Pref}^{\oplus}{}_{S,[\max(c',c''),\min(d',d'')]}$ defined as

$\forall v \in [\max(c', c''), \min(d', d'')]$ ${Pref}^{\oplus}{}_{S,[\max(c',c''),\min(d',d'')]}(v) = {Pref'}_{S,[c',d']}(v) \oplus^{pref} {Pref''}_{S,[c'',d'']}(v)$. ■

Our extend operator $\odot^{PB}$ combines the admissibility intervals by summing their endpoints (like the $\odot^{B}$ operator). As regards preferences, for each value v in the resulting admissibility interval $[c' + c'', d' + d'']$, it evaluates the new preference as follows: (i) it considers each pair of distances <*w,z*> such that $v = w + z$ and $w \in [c', d']$, $z \in [c'', d'']$, and it combines the preferences of w (in the first P_BoD) and of z (in the second P_BoD) through the $\oplus^{pref}$ operator (i.e., it evaluates $Pref'_{S,[c',d']}(w) \odot^{pref} Pref''_{S,[c'',d'']}(z)$); (ii) the resulting preferences are "merged" through the $\odot^{pref}$ operator.

Definition 10. Extend ($\odot^{PB}$). Given two P_BoDs $C' = \langle x, y, [c', d'], Pref'_{S,[c',d']} \rangle$ and $C'' = \langle y, z, [c'', d''], Pref''_{S,[c'',d'']} \rangle$, $C' \odot^{PB} C'' = \langle x, y, [c' + c'', d' + d''], Pref^{\odot}_{S,[c'+c'',d'+d'']} \rangle$,
with $Pref^{\odot}_{S,[c'+c'',d'+d'']}$ defined as
$Pref^{\odot}_{S,[c'+c'',d'+d'']}(v) = \odot^{pref}_{v=w+z \wedge w \in [c',d'] \wedge z \in [c'',d'']}(Pref'_{S,[c',d']}(w) \oplus^{pref} Pref''_{S,[c'',d'']}(z))$. ■

In the following, we term PC_GP (Path Consistency with General Preferences) our instantiation of the path-consistency algorithm of Figure 2, instantiating the operators $\oplus$ and $\odot$ with the operators $\oplus^{\mathrm{PB}}$ and $\odot^{\mathrm{PB}}$ above. Notably, it is not a specific instantiation, but a family of instantiations, depending on the SLP and on the chosen operators $\oplus^{\mathrm{pref}}$ and $\odot^{pref}$ to combine preferences. We assume that, as soon as an inconsistency is detected, PC_GP stops and reports the inconsistency.

Since our operators work in the standard way on the admissibility intervals, the following property trivially holds.

Property 1. PC_GP evaluates the minimal network of the distances in a set of P_BoDs. ■

Indeed, in the degenerate case in which the chosen SLP has a unique value p (and supposing that $\oplus^{\mathrm{pref}}$ and $\odot^{\mathrm{pref}}$ are trivially defined in such a way that $(p \oplus^{\mathrm{pref}} p) = (p \odot^{\mathrm{pref}} p) = p$), our approach trivially corresponds to the application of path-consistency to standard BoD constraints.

The complexity of the evaluation of $\oplus^{\mathrm{PB}}$ and $\odot^{\mathrm{PB}}$ is discussed below.

Complexity of $\oplus^{\mathrm{PB}}$ and $\odot^{PB}$. In the complexity analysis of the operators, we assume that the computational complexities of the operators $\oplus^{\mathrm{pref}}$ and $\odot^{\mathrm{pref}}$ are $O(T_{\oplus^{pref}})$ and $O(T_{\odot^{pref}})$, respectively[7].
If we assume that the domain $\mathbb{D}$ is countable (for example it is the domain of the integers in $\mathbb{Z}$), the complexity of $\oplus^{\mathrm{PB}}$ is $\mathrm{O}(d \cdot T_{\oplus^{pref}})$ and the complexity of $\odot^{PB}$ is $\mathrm{O}(d^2 \cdot \max(T_{\odot^{pref}}, T_{\oplus^{pref}}))$, where d is the cardinality of D, the domain of the preference function. In fact, $\oplus^{\mathrm{PB}}$ must examine all values in D, which are $\Theta(d)$, and apply the $\oplus^{\mathrm{pref}}$ operator to each one, and $\odot^{PB}$ must consider each pair of addends, which are $\Theta(d^2)$, and apply the $\oplus^{\mathrm{pref}}$ and $\odot^{\mathrm{pref}}$ operators.

[7] These operators could be computed in constant time. For instance, the user can provide a tabular definition of such operations, stating, for each pair of preferences p_i and p_j in the scale, the value of $p_i \oplus^{pref} p_j$ and of $p_i \odot^{pref} p_j$.

Notably, if the domain $\mathbb{D}$ is uncountable (for example it is the domain of the real numbers in $\mathbb{R}$), $\bigoplus^{\mathrm{PB}}$ and $\bigodot^{\mathrm{PB}}$ are not effectively computable for cardinality reasons. ■

As in the general case (see the discussion in Section 3.2), even in case the domain $\mathbb{D}$ is countable, we cannot grant the termination of PC_GP. In our approach, this is due to the fact that, informally speaking, no quiescence is necessarily reached in the general case in which no assumption can be made on the operators $\bigoplus^{pref}$ and $\bigodot^{pref}$ provided in input by the user. For example, it is possible that the operation on preferences, e.g., alternatively increases and decreases them (with respect to the total ordering in the scale) and no fixed point is reached. However, simple and "natural" restrictions on the definitions of the $\bigoplus^{\mathrm{pref}}$ operator such as the "boundedness" property below are sufficient to grant the termination of PC_GP (in case the domain $\mathbb{D}$ of differences is discrete).

Definition 11. **Boundedness.** We say that the $\bigoplus^{\mathrm{pref}}$ operator defined over a SLP S is *upper bounded* if $\forall p_1, p_2 \in S, (p_1 \bigoplus^{pref} p_2) \leq \min(p_1, p_2)$ and they are *lower bounded* if $\forall p_1, p_2 \in S, (p_1 \bigoplus^{pref} p_2) \geq \max(p_1, p_2)$ where min and max are the obvious operators, based on the total ordering of the SLP scale S. ■

Property 2. The PC_GP algorithm, when applied to the operators $\bigoplus^{\boldsymbol{PB}}$ and $\bigodot^{\boldsymbol{PB}}$, to a discrete domain $\mathbb{D}$ for differences, and to *upper or lower bounded* operator $\bigoplus^{pref}$, terminates.

Proof. Let us define the order relations $\preccurlyeq_{P\_BoD}$ and $\prec_{P\_BoD}$ between two P_BoDs and the order relations $\preccurlyeq_{Set\_P\_BoD}$ and $\prec_{Set\_P\_BoD}$ between two sets of P_BoDs.

Definition 12. Given two P_BoDs $C' = \langle x, y, [c,d], {Pref'}_{S,[c,d]} \rangle$ and $C'' = \langle x, y, [c,d], {Pref''}_{S,[c,d]} \rangle$ over the same AdI [c,d], $C' \preccurlyeq_{P\_BoD} C''$ if and only if for each value $v \in [c,d]$, ${Pref'}_{S,[c,d]}(v) \leq {Pref'}_{S,[c,d]}(v)$.

Definition 13. Given two P_BoDs $C' = \langle x, y, [c,d], {Pref'}_{S,[c,d]} \rangle$ and $C'' = \langle x, y, [c,d], {Pref''}_{S,[c,d]} \rangle$ over the same AdI [c,d], $C' \prec_{P\_BoD} C''$ if and only $C' \preccurlyeq_{P\_BoD} C''$ and there exists a value $v \in [c,d]$ such that ${Pref'}_{S,[c,d]}(v) < {Pref'}_{S,[c,d]}(v)$.

Definition 14. Given two sets B' and B'' of P_BoDs over the same set of variables X, $B' \preccurlyeq_{Set\_P\_BoD} B''$ if and only if for each pair of variables $x, y \in X$, indicating with $C'_{x,y}$ the constraint between x and y in B' and $C''_{x,y}$ the constraint between x and y in B'', $C'_{x,y} \preccurlyeq_{P\_BoD} C''_{x,y}$.

Definition 15. Given two sets B' and B'' of P_BoDs over the same set of variables X, $B' \prec_{Set\_P\_BoD} B''$ if and only if $B' \preccurlyeq_{Set\_P\_BoD} B''$ and there exists a pair of variables $x, y \in X$ such that $C'_{x,y} \prec_{P\_BoD} C''_{x,y}$.

After the first iteration of the outer loop of PC_GP, the AdIs over which the P_BoDs are defined do not change anymore since they have reached the state of minimal network (see Property 1). Thus, the next iterations can only change the preference values associated with the values in the AdIs.

If the boundedness property holds, since at each iteration the operator $\bigoplus^{pref}$ is applied, we have that, if B_i is the set of P_BoDs obtained after the i^{th} iteration of the outer loop, $B_i \preccurlyeq_{Set\_P\_BoD} B_{i+1}$. Thus, either $B_i = B_{i+1}$ and the fixed point has been reached and the algorithm terminates, or $B_i \prec_{Set\_P\_BoD} B_{i+1}$ and, since the SLP scale S has a finite number of elements the algorithm eventually terminates. ■

Now we have a sufficient condition for the termination of the PC_GP algorithm and we can analyse its complexity.

Complexity of the PC_GP algorithm. The PC_GP algorithm, when applied to the operators $\oplus^{PB}$ and $\odot^{PB}$, to a discrete domain $\mathbb{D}$ for differences, and to *bounded* operator $\oplus^{\text{pref}}$, has complexity $O(r \cdot d \cdot |V|^5 \cdot \max(T_{\oplus^{PB}}, T_{\odot^{PB}}))$, where r is the cardinality of the SLP S, d is the cardinality of $\mathbb{D}$, V are the variables, and $T_{\oplus^{PB}}$ and $T_{\odot^{PB}}$ are the complexities of the resume and extend operations, respectively. This is motivated by the consideration that the inner loop, which is $O(|V|^3 \cdot \max(T_{\oplus^{PB}}, T_{\odot^{PB}}))$, can be repeated $O(r \cdot d \cdot |V|^2)$ times, since at each iteration any single preference value of any value for any constraint can be decreased.

4 "PYRAMID" PREFERENCES

In Section 3 above, we have considered the most general case: any preference function can be used to associate preferences with BoD constraints, and any user-defined function can be used to combine preferences. The high computational complexity of the constraint propagation algorithm is a natural consequence of such a generality. Nevertheless, in many fields and applications, preference values distribute in a more regular way: they assume a maximum value at a maximum point and they decrease monotonically while moving away from such a maximum point [12]. For instance, the Gaussian distribution of preferences (over distances) is a typical example. In case preferences are layered (so that they can be expressed as a SLP), such a distribution leads to a *pyramid of nested admissibility intervals*, where the top interval has the highest preference while the bottom interval has the lowest one. This is the case, for instance, of many medical applications (consider, e.g., the running example in this paper). Other examples are vehicle routing problems, or the management of transport network.

The vehicle routing problems with soft time windows is a realistic generalization of the Travel Salesman Problem – TSP – considering multiple vehicles and soft constraints on time arrivals. In particular, Vehicle Routing Problem with Soft Time Windows (VRPSTW) is an extension of the vehicle routing problem where for each customer the service (e.g., food delivery) must be provided within a given time interval with penalty costs for early and late servicing [59]. In some specific formalizations of the VRPSTW problem, the soft constraints on the delivery times can be seen as a set of delivery time intervals with an associated preference value. Other scheduling problems, such as the management of transport networks [47], where each vehicle (e.g., a train) should arrive in each site (e.g., rail station) in a specific time, could be modeled by using pyramid preferences. A peculiar category of scheduling problems that could be modeled by using pyramid preferences involves problems where the preferences arise from the human judgment (see, e.g., [7]).

In the rest of the section, we focus on *nested* preferences, showing that, with nested preference distributions, also uncountable domains $\mathbb{D}$ for distances can be considered, and that such distributions can be exploited to achieve a more efficient propagation of the constraints based on a new definition of the $\oplus$ and $\odot$ operators.

4.1 BoD with Pyramid preferences (PyP_BoD)

The following definition formalizes the notion of preferences forming a pyramid of preferences. Given an AdI D, the preference function can be represented by a pyramid if there is at least a maximum point $v \in D$ such that preferences are monotonically increasing on the left of *v*, and are monotonically decreasing on the right of *v*.

Definition 16. Pyramid preference function (PPF). A pyramid preference function $Pref_{S_r,D}$ over the domain $D \in \mathbb{A}^{\mathbb{D}}$ with scale $S_r \in \mathbb{S}$ is a total function $Pref_{S_r,D}: D \to S_r$ such that:

$$\exists v \in D, \forall v_1, v_2 \in D,$$

$$\Big((v_1 \leq v \wedge v_2 \leq v \wedge v_1 \leq v_2) \Rightarrow Pref_{S_r,D}(v_1) \leq Pref_{S_r,D}(v_2)\Big) \wedge$$

$$\Big((v \leq v_1 \wedge v \leq v_2 \wedge v_1 \leq v_2) \Rightarrow Pref_{S_r,D}(v_1) \geq Pref_{S_r,D}(v_2)\Big)$$

If p_s is the maximum of a PPF, we say the PPF has height s.

Let $\mathbb{PP}^{\mathbb{D}}$ be the domain of PPFs. BoD with pyramid preferences is defined as following.

Definition 17. BoD with Pyramid preferences (PyP_BoD). Given a scale $S_r \in \mathbb{S}$, a PyP_BoD is a constraint $\langle x, y, [c, d], Pref_{S,[c,d]}\rangle$ of the form $\mathbb{V} \times \mathbb{V} \times \mathbb{A}^{\mathbb{D}} \times \mathbb{PP}^{\mathbb{D}}$. ■

Terminology. We call the *height* of PyP_BoD as the *height* of its preference function.

By exploiting the fact that preferences form a pyramid of height s of nested AdIs, it is possible to define a compact representation of PyP_BoD of height s.

Definition 18. Compact representation of BoD with Pyramid preferences. Given a scale $S_r \in \mathbb{S}$ of cardinality r, a PyP_BoD of height s (s≤r) can be represented in a compact way by a constraint $\langle x, y, \langle\langle [c_1, d_1], p_1\rangle, \dots, \langle [c_s, d_s], p_s\rangle\rangle\rangle$ of the form $\mathbb{V} \times \mathbb{V} \times (\mathbb{A}^{\mathbb{D}} \times \mathbb{S})^s$ in which $c_i \leq c_{i+1} \wedge d_i \geq d_{i+1}$, $1 \leq i < s$ and $p_1, \dots, p_s$ are the s lowest values in the scale S_r (i.e., $p_i = S_r(i)$, $1 \leq i \leq s$). ■

Let $\mathbb{PyP\_BoD}$ be the domain of the PyP_BoDs as defined above.

In the definition, for the sake of clarity, the preference value of each AdI is made explicit[8]. The semantics of a PyP_BoD of the form $\langle x, y, \langle\langle [c_1, d_1], p_1\rangle, \dots, \langle [c_s, d_s], p_s\rangle\rangle\rangle$ over a scale $S_r \in \mathbb{S}$ of cardinality r (s≤r) corresponds to a constraint $\langle x, y, [c_1, d_1], Pref_{S,[c,d]}\rangle$ where $Pref_{S,[c,d]}$ is such that:

(i) $\forall v \in [c_s, d_s]\ Pref_{S,D}(v) = S_r(s)$

(ii) $\forall i\ 1 \leq i < s\ \forall v \in ([c_i, d_i] - [c_{i+1}, d_{i+1}])\ Pref_{S,D}(v) = S_r(i)$.

The intuitive meaning of a PyP_BoD $\langle x, y, \langle\langle [c_1, d_1], p_1\rangle, \dots, \langle [c_s, d_s], p_s\rangle\rangle\rangle$ over a scale S_r (s≤r) is that the *difference* $y - x$ between *y* and *x* is in $[c_s, d_s]$ with preference $S_r(s)$, or in $[c_{s-1}, d_{s-1}] - [c_s, d_s]$ with preference S_r(s-1), or …. or in $[c_1, d_1] - [c_2, d_2]$ with preference $S_r(1)$.

Example 4(c). For instance, the PyP_BoD constraint $\langle NA1, NA2, \langle\langle [10,15], low\rangle, \langle [11,13], medium\rangle, \langle [12,12], high\rangle\rangle\rangle$ represents the constraint between the two successive administrations of nalidixic acid NA1 and NA2 of Example 4, i.e. the fact that the delay between two successive administrations of nalidixic acid should be of 12 hours (with a "high" preference), of 11-13 hours (with a "medium" preference) or of 10-15 hours (with a "low" preference).

As for general BoDs, also a set of PyP_BoD has a corresponding graph representation.

[8] Notably, it might be made implicit, since the preference of $[c_i,d_i]$ (i.e., of the i^{th} AdI in a PAI) is p_i (denoting S(i) the i^{th} value in the scale S).

Definition 19. Graph of PyP_BoD (PyP_BoD_G). Given an SLP S_r of cardinality r, a set B of PyP_BoD $\langle x, y, \langle\langle [c_1, d_1], p_1\rangle, \ldots, \langle [c_s, d_s], p_s\rangle\rangle\rangle$ can be represented by an oriented graph $G = \langle V, E\rangle$ with a labelling function λ, where the set of nodes V represents the set of variables in B, E ⊆ V×V, and $\lambda: E \rightarrow \mathbb{PyP\_BoD}$. ■

Example 4(d). For instance, Figure 3 shows the graph of BoD with preferences (PyP_BoD_G) representing Example 4. In the figure, each node represents a variable of the original example. RT represents the reference time (in our example, 12 of the current day); Bs, Be, Ds, De are, respectively, the breakfast and the dinner starting and ending points; NA1 and NA2 are the time points of the nalidixic acid administrations and CC1 and CC2 are the time points of the two calcium carbonate administrations. Edges in the graph represent the constraints (i.e., the PyP_BoDs) between the variables. ■

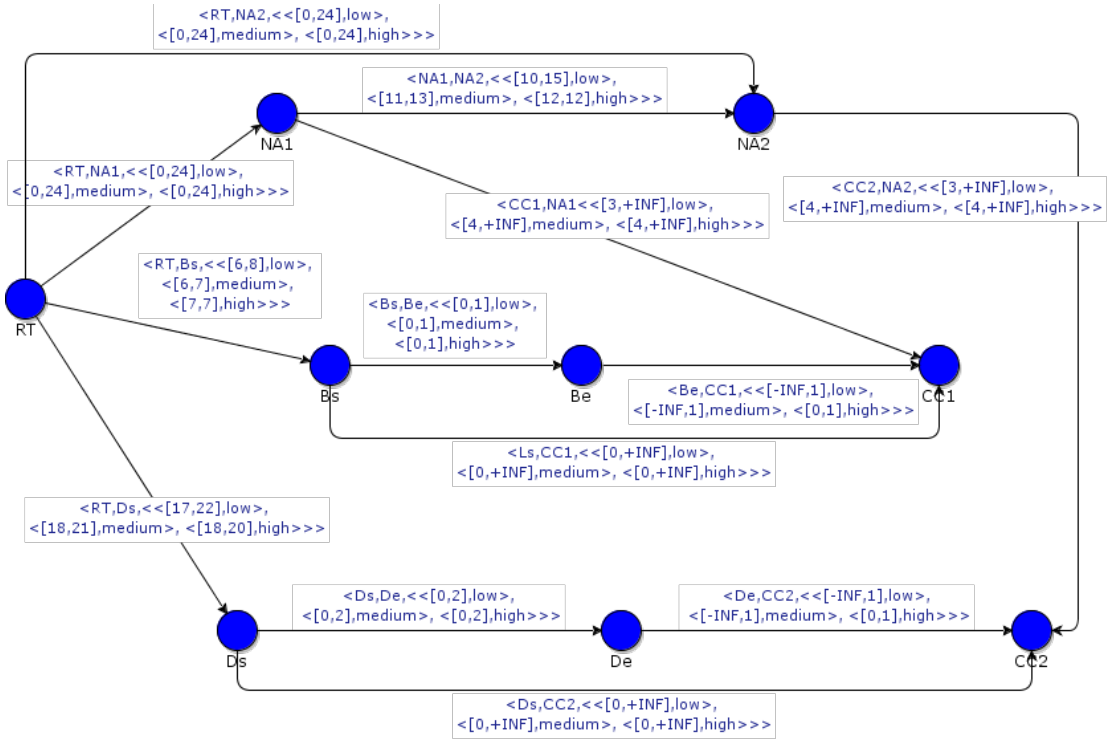

Figure 3. The PyP_BoD_G representing Example 4.

4.2 Propagating PyP_BoDs

As shown above, *nested* preferences support a more compact representation of preferences. Indeed, instead of associating a preference with each distance value in an AdI, we can discretise preferences into a finite set of nested AdIs. This fact does not only discretise the representation of dense domains of distances, but supports also a more efficient combination of preferences, since intervals of distances having the same preferences can be treated as "primitive" entities within the constraint propagation algorithms. Such a goal can be achieved through the introduction of a new definition of the $\oplus$ and $\odot$ operators, i.e., $\oplus^{\mathrm{PyPB}}$ and $\odot^{\mathrm{PyPB}}$, specialized to cope with nested preference distributions. Specifically, in the definition below, we adopt for PyP_BoDs the compact representation presented in Definition 18.

Notably, as in the general approach described in Section 3 above, the definitions of $\oplus$ and $\odot$ operators are still parametric with respect to the SLP scale adopted, and with respect to the operators $\oplus^{\text{pref}}$ and $\odot^{\text{pref}}$ on preferences. As in Section 3, to grant the termination of the propagation algorithm, we impose that the operators are *bounded*. Obviously, we also have to assume that the $\oplus^{\text{pref}}$ and $\odot^{\text{pref}}$ operators support a definition of $\oplus^{\text{PyPB}}$ and $\odot^{\text{PyPB}}$ that is *closed* with respect to the PPFs (i.e., we assume that $\oplus^{\text{pref}}$ and $\odot^{\text{pref}}$ are defined in such a way that the combination of two pyramids is still a pyramid).

The resume $\oplus^{\text{PyPB}}$ and extend $\odot^{\text{PyPB}}$ operators exploit the representation in form of pyramid to more compactly compute the results of the operators. Both operators proceed roughly in the following way:

1) they split the admissibility intervals of each PyP_BoD into fragments (the fragments are formed using the endpoints of the admissibility intervals of the operands – this step is done by the *fragments* function for $\oplus^{\text{PyPB}}$ and the *comp_fragments* function for $\odot^{\text{PyPB}}$),
2) they combine such fragments using the user-defined $\oplus^{\text{pref}}$ and $\odot^{\text{pref}}$ operators (this step is done by the $F^{\oplus}$ function for $\oplus^{\text{PyPB}}$ and by the $F^{\odot}$ function for $\odot^{\text{PyPB}}$),
3) they obtain back the syntactical form of a PyP_BoD combining the fragments and their resulting preference values (this step is done by the union operator both in $\oplus^{\text{PyPB}}$ and $\odot^{\text{PyPB}}$).

Regarding the $\oplus^{\text{PyPB}}$ operator, the function $fragments([c,d],\{[c_1,d_1],\dots,[c_k,d_k]\})$ accepts an interval [c,d] and a set of intervals. It splits [c,d] in a set of intervals that meet (in the sense of Allen [2], i.e., the ending point of an interval is the starting point of the next interval) having c, d, c_1, d_1, …, c_k, d_k as endpoints. In the appendix we report an algorithm implementing the fragments function.

Example 4(e). $fragments([10,39],\{[10,39],[11,37],[12,36]\}) = \{[10,11],[11,12],[12,36],[36,37],[37,39]\}$.

The function $F^{\oplus}$ accepts as input two PyP_BoDs. It applies the fragment function to the AdIs of the two PyP_BoDs. For each fragment returned by the function, it considers the preference value of the fragment in the two input PyP_BoDs and uses the $\oplus^{\text{pref}}$ function to compute its preference value. Thus, $F^{\oplus}(\text{C}',\text{C}'')$ returns a set of pairs $\langle[\text{c},\text{d}],\text{p}\rangle$ where the intervals are non-overlapping, cover the resulting interval and are associated with a preference value. In the $F^{\oplus}$ function, we denote by pref([c,d],C) the (maximum) preference of the AdI [c,d] in the PyP_BoD C.

Definition 21. $F^{\oplus}$ **function.** Given a scale S_r, and given two PyP_BoDs $C' = \langle x, y, \langle\langle[c'_1,d'_1],p_1\rangle,\dots,\langle[c'_k,d'_k],p_k\rangle\rangle\rangle$ and $\text{C}'' = \langle x, y, \langle\langle[c''_1,d''_1],p_1\rangle,\dots,\langle[c''_l,d''_l],p_l\rangle\rangle\rangle$ (with $1 \le k \le r$ and $1 \le l \le r$),

$F^{\oplus}(\text{C}',\text{C}'') = \{\ \langle[\text{s},\text{e}],\text{p}\rangle \mid [\text{s},\text{e}] \in \text{fragments}([c'_1,d'_1] \cap [c''_1,d''_1],\{[c'_1,d'_1],\dots,[c'_k,d'_k],[c''_1,d''_1],\dots,[c''_l,d''_l]\}) \wedge \text{p} = (\text{pref}([\text{s},\text{e}],\text{C}') \oplus^{\text{pref}} \text{pref}([\text{s},\text{e}],\text{C}''))\ \}$

Example 4(f). $F^{\oplus}(\langle NA1, NA2, \langle\langle[10,39], low\rangle,\langle[11,37], medium\rangle,\langle[12,36], high\rangle\rangle\rangle,$
$\langle NA1, NA2, \langle\langle[0,24], low\rangle,\langle[0,24], medium\rangle,\langle[0,24], high\rangle\rangle\rangle) =$
$\{\langle[10,11], low \oplus^{\text{pref}} high\rangle,\langle[11,12], medium \oplus^{\text{pref}} high\rangle,\langle[12,24], high \oplus^{\text{pref}} high\rangle\}\}$

Finally, the resume operator $\oplus^{\mathrm{PyPB}}$ performs the union of the fragments at the various preference levels in order to have a PyP_BoD as an output.

Definition 22. Resume ($\oplus^{\mathbf{PyPB}}$). Given a scale S_r and given two PyP_BoDs $C' = \langle x, y, \langle\langle [c'_1, d'_1], p_1\rangle, \dots, \langle [c'_k, d'_k], p_k\rangle\rangle\rangle$ and $C'' = \langle x, y, \langle\langle [c''_1, d''_1], p_1\rangle, \dots, \langle [c''_l, d''_l], p_l\rangle\rangle\rangle$ (with $1 \le k \le r$ and $1 \le l \le r$),
$C' \oplus^{\mathrm{PyPB}} C'' = \langle x, y, \langle\langle [c_i, d_i], p_i\rangle, i = 1, \dots, r \mid [c_i, d_i] = \bigcup_{\langle [s,e], p\rangle \in F^{\oplus}(C', C'') \wedge p \ge p_i} [s, e] \wedge [c_i, d_i] \neq \emptyset\rangle\rangle$.

Example 4(g). As an example, we compute $\left\langle NA1, NA2, \langle\langle [10,39], low\rangle, \langle [11,37], medium\rangle, \langle [12,36], high\rangle\rangle\right\rangle \oplus^{\mathrm{PyPB}}$ $\langle NA1, NA2, \langle\langle [0,24], low\rangle, \langle [0,24], medium\rangle, \langle [0,24], high\rangle\rangle\rangle$ assuming that the $\oplus^{\mathrm{pref}}$ operator is the *min* function (with $low < medium < high$). Since $F^{\oplus}(\langle NA1, NA2, \langle\langle [10,39], low\rangle, \langle [11,37], medium\rangle, \langle [12,36], high\rangle\rangle\rangle$, $\langle NA1, NA2, \langle\langle [0,24], low\rangle, \langle [0,24], medium\rangle, \langle [0,24], high\rangle\rangle\rangle) =$
$\{\{\langle [10,11], low \oplus^{\mathrm{pref}} high\rangle, \langle [11,12], medium \oplus^{\mathrm{pref}} high\rangle, \langle [12,24], high \oplus^{\mathrm{pref}} high\rangle\}\}$, for obtaining the AdI for the preference level low, the $\oplus^{\mathrm{PyPB}}$ operator performs the union $\cup\{[10,11], [11,12], [12,24]\}$, and for obtaining the AdI for the preference level $medium$, it performs the union $\cup\{[11,12], [12,24]\}$. Finally, to obtain the AdI for the preference level $high$, it performs the singleton union $\cup\{[11,12], [12,24]\}$. Thus, $\left\langle NA1, NA2, \langle\langle [10,39], low\rangle, \langle [11,37], medium\rangle, \langle [12,36], high\rangle\rangle\right\rangle \oplus^{\mathrm{PyPB}}$
$\left\langle NA1, NA2, \langle\langle [0,24], low\rangle, \langle [0,24], medium\rangle, \langle [0,24], high\rangle\rangle\right\rangle =$
$\langle NA1, NA2, \langle\langle [10,24], \mathrm{low}\rangle, \langle [11,24], medium\rangle, \langle [12,24], high\rangle\rangle\rangle$.

Regarding the $\odot^{\mathrm{PyPB}}$ operator, the function $comp\_fragments(\langle [c_1, d_1], \dots, [c_k, d_k]\rangle, \langle [c'_1, d'_1], \dots, [c'_l, d'_l]\rangle)$ accepts as arguments two sets of nested AdIs, and returns a set of non-overlapping intervals that have as endpoints the sum of the endpoints of the intervals passed as arguments (considering each interval resulting from the first argument paired with each interval resulting from the second argument).

Definition 23. comp_fragments function. $\mathrm{comp\_fragments}(\langle [s_1, e_1], \dots, [s_k, e_k]\rangle, \langle [s'_1, e'_1], \dots, [s'_l, e'_l]\rangle) =$ $fragments([s_1 + s'_1, e_1 + e'_1], \{[s + s', e + e'] \mid [s, e] \in fragments([s_1, e_1], \{[s_1, e_1], \dots, [s_k, e_k]\}) \wedge [s', e'] \in fragments([s'_1, e'_1], \{[s'_1, e'_1], \dots, [s'_l, e'_l]\})\})$.

Example 4(h). Since $fragments([0,24], \{[0,24], [0,24], [0,24]\}) = \{[0,24]\}$ and
$fragments([10,15], \{[10,15], [11,13], [12,12]\}) = \{[10,11], [11,12], [12,12], [12,13], [13,15]\}$,
$comp\_fragments(\langle [0,24], [0,24], [0,24]\rangle, \langle [10,15], [11,13], [12,12]\rangle) =$
$fragments([10,39], \{[10,35], [11,36], [12,36], [12,37], [13,39]\}) =$
$\{[10,11], [11,12], [12,12], [12,13], [13,35], [35,36], [36,37], [37,39]\}$.

The function $F^{\odot}(C', C'')$ computes the preference value for each of the fragments [s,e] returned by *comp_fragments*. The preference value is determined by considering the preference values of the fragments of C' and C'' whose composition contains [s,e] (by construction, the fragments returned by *comp_fragments* are always contained in the

fragments of C' and of C''). Thus, for each pair of fragments $[s_i', e_i']$ of C' and $[s_j'', e_j'']$ of C'' such that $[s_i'+s_j'', e_i'+e_j'']$ contains a fragment [s,e] in *comp_fragments*, $F^{\odot}(C', C'')$ first computes the preference value of the combination of the two preference values holding together using the $\oplus^{pref}$ operator and then – to determine the preference value of [s,e] – it merges these preference values using the $\odot^{pref}$ operator.

Definition 24. $F^{\odot}$ **function.** Given a scale S_r, and given two PyP_BoDs $C' = \langle x, y, \langle\langle [c_1', d_1'], p_1\rangle, \ldots, \langle [c_k', d_k'], p_k\rangle\rangle\rangle$ and $C'' = \langle x, y, \langle [c_1'', d_1''], p_1\rangle, \ldots, \langle [c_l'', d_l''], p_l\rangle\rangle\rangle$ (with $1 \le k \le r$ and $1 \le l \le r$),

$F^{\odot}(C', C'') = \{ \langle [s, e], p\rangle \mid [s, e] \in \text{comp\_fragments}(\langle [c_1', d_1'], \ldots, [c_k', d_k']\rangle, \langle [c_1'', d_1''], \ldots, [c_l'', d_l'']\rangle) \wedge$

$$p = \odot^{pref}_{\substack{[s_i', e_i'] \in \text{fragments}([c_1', d_1'], \{[c_1', d_1'], \ldots, [c_k', d_k']\}) \wedge \\ [s_j'', e_j''] \in \text{fragments}([c_1'', d_1''], \{[c_1'', d_1''], \ldots, [c_l'', d_l'']\}) \wedge \\ [s_i'+s_j'', e_i''+e_j''] \supseteq [s,e]}} \{\text{pref}([s_i', e_i'], C') \oplus^{pref} \text{pref}([s_j'', e_j''], C'')\}.$$

Example 4(i).

$F^{\odot}(\langle\langle [0,24], low\rangle, \langle [0,24], medium\rangle, \langle [0,24], high\rangle\rangle, \langle\langle [10,15], low\rangle, \langle [11,13], medium\rangle, \langle [12,12], high\rangle\rangle) =$

$\langle\langle [10,11], low \oplus^{\text{pref}} high\rangle,$

$\langle [11,12], (low \oplus^{\text{pref}} high) \odot^{pref} (medium \oplus^{\text{pref}} high)\rangle,$

$\langle [12,12], (low \oplus^{\text{pref}} high) \odot^{pref} (medium \oplus^{\text{pref}} high) \odot^{pref} (high \oplus^{\text{pref}} high)\rangle,$

$\langle [12,13], (low \oplus^{\text{pref}} high) \odot^{pref} (medium \oplus^{\text{pref}} high) \odot^{pref} (high \oplus^{\text{pref}} high)\rangle,$

$\langle [13,35], (low \oplus^{\text{pref}} high) \odot^{pref} (medium \oplus^{\text{pref}} high) \odot^{pref} (high \oplus^{\text{pref}} high)\rangle,$

$\langle [35,36], (low \oplus^{\text{pref}} high) \odot^{pref} (medium \oplus^{\text{pref}} high) \odot^{pref} (high \oplus^{\text{pref}} high)\rangle,$

$\langle [36,37], (low \oplus^{\text{pref}} high) \odot^{pref} (medium \oplus^{\text{pref}} high)\rangle,$

$\langle [37,39], low \oplus^{\text{pref}} high\rangle$

Finally, as the resume $\oplus^{\text{PyPB}}$ operator, also the extension operator $\odot^{\text{PyPB}}$ performs the union of the fragments returned by $F^{\odot}$ at the various preference levels in order to have a PyP_BoD as an output.

Definition 25. Extension ($\odot^{\text{PyPB}}$). Given a scale S_r, and given two PPCs $C' = \langle x, y, \langle\langle [c_1', d_1'], p_1\rangle, \ldots, \langle [c_k', e_k'], p_k\rangle\rangle\rangle$ and $C'' = \langle y, z, \langle\langle [c_1'', d_1''], p_1\rangle, \ldots, \langle [c_l'', d_l''], p_l\rangle\rangle\rangle$ (with $1 \le k \le r$ and $1 \le l \le r$), their extension is

$$C' \odot^{\text{PyPB}} C'' = \langle x, z, \langle\langle [c_i, d_i], p_i\rangle, i = 1, \ldots, r \mid [c_i, d_i] = \bigcup_{\langle [s,e], p\rangle \in F^{\odot}(C', C'') \wedge p \ge p_i} [s, e] \wedge [c_i, d_i] \neq \emptyset\rangle\rangle$$

Example 4(j). As an example, we compute $\left\langle RT, NA1, \langle\langle [0,24], low\rangle, \langle [0,24], medium\rangle, \langle [0,24], high\rangle\rangle\right\rangle \odot^{\text{PyPB}}$ $\left\langle NA1, NA2, \langle\langle [10,15], low\rangle, \langle [11,13], medium\rangle, \langle [12,12], high\rangle\rangle\right\rangle$ assuming that the $\oplus^{\text{pref}}$ operator is the *min* function and that the $\odot^{pref}$ is the max function. Since, $F^{\odot}(\langle\langle [0,24], low\rangle, \langle [0,24], medium\rangle, \langle [0,24], high\rangle\rangle, \langle\langle [10,15], low\rangle, \langle [11,13], medium\rangle, \langle [12,12], high\rangle\rangle) =$ $\langle\langle [10,11], low\rangle, \langle [11,13], medium\rangle, \langle [12,12], high\rangle, \langle [12,13], high\rangle, \langle [13,35], high\rangle, \langle [35,36], high\rangle, \langle [36,37], medium\rangle,$ $\langle [37,39], low\rangle\rangle$, for obtaining the AdI for the preference level low, the $\odot^{\text{PyPB}}$ operator performs the union $\cup\{[10,11], [11,12], [12,12], [12,13], [13,35], [35,36], [36,37], [37,39]\}$, for obtaining the AdI for the preference

level $medium$, it performs the union $\bigcup\{[11,12],[12,12],[12,13],[13,35],[35,36],[36,37]\}$, and, for obtaining the AdI for the preference level $high$, it performs the union $\bigcup\{[12,12][13,35][35,36]\}$. Thus, $\left\langle RT, NA1, \left\langle \langle [0,24], low\rangle, \langle [0,24], medium\rangle, \langle [0,24], high\rangle \right\rangle \right\rangle \odot^{\mathrm{PyPB}}$
$\left\langle NA1, NA2, \left\langle \langle [10,15], low\rangle, \langle [11,13], medium\rangle, \langle [12,12], high\rangle \right\rangle \right\rangle =$
$\langle RT, NA2, \langle \langle [10,39], low\rangle, \langle [11,37], medium\rangle, \langle [12,36], high\rangle \rangle\rangle$.

Complexity. Regarding the $\oplus^{\mathrm{PyPB}}$ operator, the *fragments* function can operate in time $\mathrm{O}(r)$, where *r* is the number of preference values in S_r, because the function $fragments([c,d],\{[c_1,d_1],\dots,[c_k,d_k]\})$ – since the endpoints are already sorted – can basically operate linearly by considering the endpoints of the AdIs passed as parameters, which are at most *4r* (since the two constraints can have at most *r* AdIs and each one has two endpoints). The $F^{\oplus}$ function invokes the fragments function and, for each of the O(r) fragments, obtains its preference value and applies the $\oplus^{\mathrm{pref}}$ operator. We assume that the preference value can be obtained in $\mathrm{O}(1)$ and we denote the complexity of the $\oplus^{\mathrm{pref}}$ operator as $\mathrm{O}(T_{\oplus^{pref}})$. Thus, the $F^{\oplus}$ function operates in time $\mathrm{O}(r \cdot T_{\oplus^{pref}})$. The $\oplus^{\mathrm{PyPB}}$ operator invokes the $F^{\oplus}$ function, which returns O(r) fragments, and for each preference value it performs the union of the fragments that have at least the given preference value. Since we assume that the fragments are already sorted, the union can be computed in linear time and, in the worst case, the union must consider *r* times the output of the $F^{\oplus}$ function, the $\oplus^{\mathrm{PyPB}}$ operator operates in $\mathrm{O}(\max(r \cdot T_{\oplus^{pref}}, r^2))$.

Regarding the $\odot^{\mathrm{PyPB}}$ operator, the *comp_fragments* function invokes the *fragments* function on the two AdIs and, considering each pair of fragments, invokes again the *fragments* function. Thus, it operates in time $\mathrm{O}(r^2)$, where *r* is the number of preference values in S_r. The $F^{\odot}$ function invokes the *comp_fragments* function and, for each pair of fragments resulting from the AdIs of the two constraints (whose sum contains a fragment returned by *comp_fragments*), it applies the $\oplus^{\mathrm{pref}}$ operator (and then the $\odot^{\mathrm{pref}}$ operator). Since *fragments* is $\mathrm{O}(r)$, *comp_fragments* is $\mathrm{O}(r^2)$ and there are $\mathrm{O}(r^2)$ pairs of fragments, the $F^{\odot}$ function in the worst case applies the $\oplus^{\mathrm{pref}}$ and $\odot^{\mathrm{pref}}$ operators once for each pair and operates in time $\mathrm{O}(r^2 \cdot T_{\oplus^{pref}} \cdot T_{\odot^{pref}})$. Finally, the $\odot^{\mathrm{PyPB}}$ operator invokes the $F^{\odot}$ function, which can return $\mathrm{O}(r^2)$ fragments, and, for each preference value, it performs the union of the fragments returned by $F^{\odot}$. Thus, in the worst case, $\odot^{\mathrm{PyPB}}$ must consider *r* times the output of the $F^{\odot}$ function, thus it operates in $\mathrm{O}(\max(r^2 \cdot T_{\oplus^{pref}} \cdot T_{\odot^{pref}}, r^3))$.

Notably, the $\oplus^{\mathrm{PyPB}}$ and $\odot^{\mathrm{PyPB}}$ operators, unlike the $\oplus^{\mathrm{PB}}$ and $\odot^{\mathrm{PB}}$ operators, can deal both with discrete and dense domains, since they operate not on the single values in $\mathbb{D}$ but on the intervals induced by the pyramid structure.

At this point, we can summarise the main points of this section. At the beginning of the section, we have defined a special form of BoDs with preferences called PyP_BoDs. Then, in order to define the extend and resume operators on PyP_BoDs, we introduced the functions $F^{\oplus}$ and $F^{\odot}$ and the notion of fragment. In the remaining part of this section, we connect the formalism presented in Section 3 and the formalism presented in this section with Property 3. Indeed, Property 3 shows that the definitions of the resume and extend operators on PyP_BoDs are a particular case of the definitions of the corresponding operators on P_BoDs. To define such a property, we first introduce an auxiliary conversion function.

We define a function $\rho^{PyPB\rightarrow PB}$ that converts a PyP_BoD into a P_BoD by following the definition of the PyP_BoDs introduced above.

Definition 26. $\boldsymbol{\rho^{PyPB\rightarrow PB}}$. Given a scale S_r and a PyP_BoD $C = \langle x, y, \langle\langle [c_1, d_1], p_1\rangle, \dots, \langle [c_k, d_k], p_k\rangle\rangle\rangle$, $\rho^{PyPB\rightarrow PB}(C)$ is a P_BoD such that

$$\rho^{PyPB\rightarrow PB}(C) = \langle x, y, [c_1, d_1], Pref_{S,[c_1,d_1]}\rangle$$

with $\forall v \in [c_k, d_k], Pref_{S,[c_1,d_1]}(v) = S_r(k)$ and $\forall i\ 1 \leq i < k\ \forall v \in ([c_i, d_i] - [c_{i+1}, d_{i+1}])\ Pref_{S,[c_1,d_1]}(v) = S_r(i)$.

Property 3. Assuming that the $\bigoplus^{pref}$ and $\bigodot^{pref}$ operators are closed with regard to PPFs, given two PyP_BoDs $C' = \langle x, y, \langle\langle [c'_1, d'_1], p_1\rangle, \dots, \langle [c'_k, d'_k], p_k\rangle\rangle\rangle$ and $C'' = \langle x, y, \langle\langle [c''_1, d''_1], p_1\rangle, \dots, \langle [c''_l, d''_l], p_l\rangle\rangle\rangle$,

$$\rho^{PyPB\rightarrow PB}(C' \oplus^{PyPB} C'') = \rho^{PyPB\rightarrow PB}(C') \oplus^{PB} \rho^{PyPB\rightarrow PB}(C'')$$

and, given two PyP_BoDs $C' = \langle x, y, \langle\langle [c'_1, d'_1], p_1\rangle, \dots, \langle [c'_k, d'_k], p_k\rangle\rangle\rangle$ and $C'' = \langle y, z, \langle\langle [c''_1, d''_1], p_1\rangle, \dots, \langle [c''_l, d''_l], p_l\rangle\rangle\rangle$,

$$\rho^{PyPB\rightarrow PB}(C' \odot^{PyPB} C'') = \rho^{PyPB\rightarrow PB}(C') \odot^{PB} \rho^{PyPB\rightarrow PB}(C'').$$

Proof. For the sake of brevity, we consider the resume operators $\bigoplus^{PyPB}$ and $\bigoplus^{PB}$; the proof for the extend operators $\bigodot^{PyPB}$ and $\bigodot^{PB}$ is developed in a similar way.

We prove the equivalence by proving that the left-hand side (henceforth, lhs) and the right-hand side (henceforth, rhs) are defined on the same pair of variables, on the same domain and that the preference functions are the same.

In the case of the resume operators, it is trivial to verify that they use the same pairs of variables.

Regarding the domain, by definition of $\bigoplus^{PyPB}$, the domain of the Pref function on the lhs is the union of all the fragments returned by the $F^{\oplus}$ function, which is (by definition of $F^{\oplus}$) the interval $[c'_1, d'_1] \cap [c''_1, d''_1]$; by definition of $\bigoplus^{PB}$, the domain of the Pref function on the rhs is the intersection between the domain of $\rho^{PyPB\rightarrow PB}(C')$ and the domain of $\rho^{PyPB\rightarrow PB}(C'')$, i.e., $[c'_1, d'_1] \cap [c''_1, d''_1]$.

Regarding the preference values, we prove that, given an arbitrary value $v \in [c'_1, d'_1] \cap [c''_1, d''_1]$ with preference value $Pref_{S,[c'_1,d'_1]\cap[c''_1,d''_1]}(v) = S_r(i)$ on the lhs, v has the same preference value on the rhs (part 1 of the proof in the following), and vice versa (part 2).

(**Part 1**) Given an arbitrary value $v \in [c'_1, d'_1] \cap [c''_1, d''_1]$ on the lhs with preference value $Pref_{S,[c'_1,d'_1]\cap[c''_1,d''_1]}(v) = S_r(i)$, by definition of $\rho^{PyPB\rightarrow PB}$, there exists a PyP_BoD $C''' = \left\langle x, y, \langle\langle [c'''_1, d'''_1] = [c'_1, d'_1] \cap [c''_1, d''_1], p_1\rangle, \dots, \langle [c'''_m, d'''_m], p_m\rangle\rangle\right\rangle$ such that $C''' = C' \oplus^{PyPB} C''$ where either a) there is an AdI $[c_i, d_i], 1 \leq i < m$ such that $v \in ([c_i, d_i] - [c_{i+1}, d_{i+1}])$ or b) $v \in [c_m, d_m]$ (and $m = i$). Let us consider the case a), the other is analogous and simpler.

By definition of $\bigoplus^{PyPB}$, $[c_i, d_i]$ is the union of all the fragments $\langle [s, e], p\rangle$ returned by $F^{\oplus}(C', C'')$ such that $p \geq p_i$. Since $v \in ([c_i, d_i] - [c_{i+1}, d_{i+1}])$, there does not exist any fragment $\langle [s, e], p_{i+1}\rangle$ returned by $F^{\oplus}(C', C'')$ such that $v \in [s, e]$ and there exists a fragment $\langle [s, e], p_i\rangle$ returned by $F^{\oplus}(C', C'')$ such that $v \in [s, e]$.

By definition of $F^{\oplus}$, there exists an interval $[s, e] \in fragments([c'_1, d'_1] \cap [c''_1, d''_1], \{[c'_1, d'_1], \dots, [c'_k, d'_k], [c''_1, d''_1], \dots, [c''_l, d''_l]\})$ such that $v \in [s, e]$ and $p_i =$

$(pref([s,e],C') \oplus^{pref} pref([s,e],C''))$ and there does not exist any interval $[s,e] \in fragments([c'_1,d'_1] \cap [c''_1,d''_1],\{[c'_1,d'_1],\dots,[c'_k,d'_k],[c''_1,d''_1],\dots,[c''_l,d''_l]\})$ such that $p_{i+1} = (pref([s,e],C') \oplus^{pref} pref([s,e],C''))$ and $v \in [s,e]$.

By definition of the fragments function, there exist an AdI $[c'_{i'},d'_{i'}]$ of C' and an AdI $[c''_{i''},d''_{i''}]$ of C'' such that $v \in [c'_{i'},d'_{i'}]$, $v \in [c''_{i''},d''_{i''}]$, and $p_i = p_{i'} \oplus^{pref} p_{i''}$ and there do not exist any AdI $[c'_{j'},d'_{j'}]$ of C' and any AdI $[c''_{j''},d''_{j''}]$ of C'' such that $v \in [c'_{j'},d'_{j'}]$, $v \in [c''_{j''},d''_{j''}]$, and $p_{j'} \oplus^{pref} p_{j''} > p_i$.

Thus, moving to the rhs, by definition of $\rho^{PyPB\to PB}$, $Pref_{S,[c'_1,d'_1]}(v) = S_r(i')$ and $Pref_{S,[c''_1,d''_1]}(v) = S_r(i'')$.

By definition of $\oplus^{PB}$, $Pref^{\oplus}{}_{S,[\max(c',c''),\min(d',d'')]}(v) = Pref'_{S,[c',d']}(v) \oplus^{pref} Pref''_{S,[c'',d'']}(v)$. Thus, $Pref^{\oplus}{}_{S,[\max(c',c''),\min(d',d'')]}(v) = S_r(i') \oplus^{pref} S_r(i'') = pref([s,e],C') \oplus^{pref} pref([s,e],C'') = p_i = S_r(i)$.

(**Part 2**) Given an arbitrary value $v \in [c'_1,d'_1] \cap [c''_1,d''_1]$ on the rhs with preference value $Pref_{S,[c'_1,d'_1]\cap[c''_1,d''_1]}(v) = S_r(i)$, by definition of $\oplus^{PB}$, there exist two P_BoDs $C^{PB'} = \langle x,y,[c',d'],Pref'_{S,[c',d']}\rangle$ and $C^{PB''} = \langle x,y,[c'',d''],Pref''_{S,[c'',d'']}\rangle$ such that $v \in [c',d']$, $v \in [c'',d'']$ and $Pref'_{S,[c',d']}(v) \oplus^{pref} Pref''_{S,[c'',d'']}(v) = S_r(i)$. Let $Pref'_{S,[c',d']}(v)$ be $p_{i'} = S_r(i')$ and $Pref''_{S,[c'',d'']}(v)$ be $p_{i''} = S_r(i'')$.

By definition of $\rho^{PyPB\to PB}$, there exist two PyP_BoDs $C' = \langle x,y,\langle\langle[c'_1,d'_1],p_1\rangle,\dots,\langle[c'_{m'},d'_{m'}],p_{m'}\rangle\rangle\rangle$ and $C'' = \langle x,y,\langle\langle[c''_1,d''_1],p_1\rangle,\dots,\langle[c''_{m''},d''_{m''}],p_{m''}\rangle\rangle\rangle$ such that, considering C', either a') there is an AdI $[c_{i'},d_{i'}], 1 \le i' < m'$ such that $v \in ([c_{i'},d_{i'}] - [c_{i'+1},d_{i'+1}])$ or b') $i' = m'$ and $v \in [c_{m'},d_{m'}]$; the same holds for C'': either a'') there is an AdI $[c_{i''},d_{i''}], 1 \le i'' < m''$ such that $v \in ([c_{i''},d_{i''}] - [c_{i''+1},d_{i''+1}])$ or b'') $i'' = m''$ and $v \in [c_{m''},d_{m''}]$. As above, we consider cases a') and a'') since the other combinations of cases (i.e., a' and b'', b' and a'', b' and b'') are analogous and simpler.

Moving to the lhs, since $v \in ([c_{i'},d_{i'}] - [c_{i'+1},d_{i'+1}])$ and $[c_{i'+1},d_{i'+1}] \subseteq [c_{i'},d_{i'}]$, either (i) $c_{i'} \le v \le c_{i'+1}$ or (ii) $d_{i'+1} \le v \le d_{i'}$, and, similarly, since $v \in ([c_{i''},d_{i''}] - [c_{i''+1},d_{i''+1}])$ and $[c_{i''+1},d_{i''+1}] \subseteq [c_{i''},d_{i''}]$, either (a) $c_{i''} \le v \le c_{i''+1}$ or (b) $d_{i''+1} \le v \le d_{i''}$. Thus, there are four cases. Let us consider the case (i)(a), thus $\max(c_{i'},c_{i''}) \le v \le \min(c_{i'+1},c_{i''+1})$. The other three cases are analogous.

Following the definition of $\oplus^{PyPB}$, we consider the function $F^{\oplus}(C',C'')$. Since $fragments([c'_1,d'_1] \cap [c''_1,d''_1],\{[c'_1,d'_1],\dots,[c'_k,d'_k],[c''_1,d''_1],\dots,[c''_l,d''_l]\})$ returns a set of non-overlapping intervals covering $[c'_1,d'_1] \cap [c''_1,d''_1]$ and having $c'_1,d'_1,\dots,c'_k,d'_k,c''_1,\dots,d''_1,\dots,c''_l,\dots,d''_l$ as endpoints, there is exactly one fragment $[s,e]$ returned by *fragments* such that $v \in [s,e]$. By definition of PyP_BoD, $c'_1 \le \dots \le c'_{k-1} \le c'_k \le d'_k \le d'_{k-1} \le \dots \le d'_1$ and $c''_1 \le \dots \le c''_{l-1} \le c''_l \le d''_l \le d''_{l-1} \le \dots \le d''_1$. Thus, $[s,e] = [\max(c_{i'},c_{i''}),\min(c_{i'+1},c_{i''+1})]$. By construction, $\mathrm{pref}([s,e],C') = p_{i'}$ and $\mathrm{pref}([s,e],C'') = p_{i''}$, and the $F^{\oplus}$ function returns the pair $\langle[s,e],p_{i'} \oplus^{pref} p_{i''}\rangle$. By construction, $p_{i'} \oplus^{pref} p_{i''} = p_i$. For the definition of $\oplus^{PyPB}$, in order to build an AdI $[c_i,d_i]$ with preference value p_i, a union is performed over all the pairs returned by $F^{\oplus}$ that have preference values higher than or equal to p_i. Since we assume that $\oplus^{pref}$ is defined in such a way that PPFs are closed under $\oplus^{PyPB}$, the intervals included between $\max(c_{i'},c_{i''}) = c_i$ and $\min(d_{i'},d_{i''}) = d_i$ are all and only the ones that have preference value higher than or equal to p_i. For the definition of $\rho^{PyPB\to PB}$, since $v \in [s,e] = [\max(c_{i'},c_{i''}),\min(c_{i'+1},c_{i''+1})]$, $Pref_{S,[c_1,d_1]}(v) = p_i$. ■

5 CLOSED SEMIRING-BASED OPERATIONS ON "PYRAMID" CONSTRAINTS

For the sake of generality, until now we have considered arbitrary (user-provided) operators for specifying how the preferences can be combined (Section 3). We now show that, if the *resume* and *extend* operators satisfy certain conditions, PyP_BoDs can be managed in a much more efficient way. To achieve such a goal, we consider a general property of a class of algorithms proposed in [18]. Indeed, the simplification of the path-consistency algorithm shown is Figure 2 obtained by removing from it the "repeat … until no constraint is changed" (i.e., removing lines 6 and 11; see Figure 4 below) exactly corresponds to Cormen et al.'s general algorithm *Compute-Summaries(λ,V,E,⊕,⊙,**0**,**1**)* in [18].

Compute-Summaries(λ,V,E,⊕,⊙,0,1) algorithm

```
1. n ← |V|
2. for i←1 to n do
3.    for j←1 to n do
4.       L_ij ← λ(i,j)
5. for k←1 to n do
6.    for i←1 to n do
7.       for j←1 to n do
8.          L_ij ← L_ij ⊕ (L_ik ⊙ L_kj)
9. return L
```

Figure 4. Cormen et al.'s general algorithm *Compute-Summaries* (adapted from [18]).

Compute-Summaries is a parametric *all-pairs shortest paths* algorithm that solves a number of problems involving directed paths in graphs. As in the case of the path consistency algorithm, also compute-summaries is parametric with respect to the labelling function $\lambda:E \rightarrow L$, the "*extension*" operator $\odot$, and the "*resume*" operator $\oplus$, as well as on two additional parameters, i.e., the *identities* for $\odot$ (indicated by $\mathbf{1}$) and for $\oplus$ (indicated by $\mathbf{0}$). Cormen et al. show that, if $\langle L,\oplus,\odot,\mathbf{0},\mathbf{1}\rangle$ is a *closed semiring*, the algorithm computes the all-pairs shortest paths, which corresponds in our context to the minimal network of the constraints. Therefore, we rely on such a result and define the *extension* $\odot^{\mathrm{P}}$ operator, the *resume* $\oplus^{\mathrm{P}}$ operator and their *identities* on the compact PyP_BOD constraints (that is the operators consider both *admissibility intervals* and *preferences*) so that $\langle L,\oplus^{P},\odot^{P},\mathbf{0},\mathbf{1}\rangle$ is a *closed semiring*. In such a way, we can compute the minimal network of the distances, and their preferences, without the need of the external loop of the algorithm in Figure 2. Obviously, this move is very advantageous from the computational point of view.

Complexity. The complexity of the algorithm Compute-Summaries is $\Theta(|V|^3 \cdot \max(T_{\oplus}, T_{\odot}))$, where V are the variables, and $T_{\oplus}$ and $T_{\odot}$ are the complexities of the resume and extend operations, respectively.

To be able to exploit Compute-Summaries to evaluate the minimal network of our PyP_BoD constraints, we cannot anymore assume that the operations $\oplus^{pref}$ and $\odot^{pref}$ on preference are arbitrary operations provided in input by the users. They have to be defined in such a way that the *extension* $\odot^{\mathrm{P}}$ and *resume* $\oplus^{\mathrm{P}}$ operators (which operate on both distances and preferences, adopting $\oplus^{pref}$ and $\odot^{pref}$ to combine preferences) can form a closed semiring.

Before moving to an example of a closed semiring, we briefly summarize and comment its definition.

Definition 27. Closed semiring [1]**.** A closed semiring is a structure $\langle S,\oplus,\odot,0,1\rangle$, where S is a set of elements, and $\oplus$ and $\odot$ are binary operations on S satisfying the properties:

- $\langle S,\oplus,0\rangle$ is a monoid (i.e., it is closed under $\oplus$, $\oplus$ is associative, and 0 is an identity). $\langle S,\odot,1\rangle$ is a monoid and 0 is an annihilator.
- $\oplus$ is commutative.
- $\odot$ distributes over $\oplus$.
- if $a_1, a_2, \ldots, a_i, \ldots$ is a countable sequence of elements in S, $a_1 \oplus a_2 \oplus \ldots \oplus a_i \oplus \ldots$ exists and is unique. Moreover, associativity, commutativity and idempotence apply to infinite as well as finite sums.
- $\odot$ must distribute over countably infinite sums as well as finite ones.

If $\langle \mathbb{PyP\_BoD}, \oplus^P, \odot^P, \bot, \top\rangle$ is a *closed semiring*, Cormen et al.'s Compute-Summaries algorithm computes the *all-pairs shortest paths* of a graph, by determining new labels for the edges through the operators resume ($\oplus$) and extension ($\odot$) [1]. As an intuition, the properties of *closed semirings* grant that the result of the application of resume and extension is independent of the order in which such operations are applied. In fact, such an independence is granted by:

(1) the *associativity property* derived by the monoids for the single operators, and

(2) the *distributive property* of the extension operator with respect to the resume operator. In this case, given two constraints C_1 and C_2 between variables *x* and *y*, and a constraint C_3 between *y* and *z*, one obtains the same result (a) by first resuming C_1 and C_2, and then by composing the result with C_3 through the extension operator to obtain the constraint between *x* and *z*, or (b) by first composing C_1 and C_2, and C_2 and C_3 through the extension operator, to obtain two new constraints between C_1 and C_3, and then combine them through the resume operator.

Any instance of *closed semiring* is feasible. In the following, we propose a concrete example, that we obtain by using the ***min*** function to ***resume*** preferences (i.e., we instantiate $\oplus^{pref}$ with the *min* function) and ***max*** to extend them (i.e., we instantiate $\odot^{pref}$ with the *max* function). Notably, the choice of such operations to combine preferences is quite widely adopted by the specialized literature (consider, e.g., [13]). In particular, such a choice is suitable for medical applications, such as the one in examples above.

We now show that, with the above choice of *min* for $\oplus^{pref}$ and *max* for $\odot^{pref}$, and assuming that the domain $\mathbb{D}$ contains also the elements 0 and ∞, we can provide a suitable (and compact) definition of the *resume* ($\oplus^{\mathrm{P}}$) and *extend* ($\odot^{\mathrm{P}}$) operators on PyP_BoDs in such a way to obtain a *closed semiring* (so that the algorithm in Figure 4 can be exploited to obtain the *minimal network* of a graph of PyP_BoD constraints).

Indeed, the above choice of the $\oplus^{pref}$ and $\odot^{pref}$ operators supports a compact and efficient definition of $\oplus^{\mathrm{P}}$ and $\odot^{\mathrm{P}}$, in which operations (on both distances and preferences) are performed "*layer-by-layer*".

Specifically, the resume operator $\oplus^{\mathrm{P}}$ computes, at each level *i* of the PyP_BoDs, the intersection (i.e., $[\max(c_i', c_i''), \min(d_i', d_i'')]$) between the AdI at level *i* in C' (i.e., $[c_i', d_i']$ in the definition of the resume operator below) and C'' (i.e., $[c_i'', d_i'']$ in the definition below), and pairs its result with the preference value p_i corresponding with that level, until it reaches the highest level for both constraints.

Definition 28. Resume ($\oplus^{P}$). Given a scale S_r, and given two PyP_BoDs $C' = \langle x, y, \langle\langle [c'_1, d'_1], p_1\rangle, \dots, \langle [c'_k, d'_k], p_k\rangle\rangle\rangle$ and $C'' = \langle x, y, \langle\langle [c''_1, d''_1], p_1\rangle, \dots, \langle [c''_l, d''_l], p_l\rangle\rangle\rangle$ (with $1 \leq k \leq r$ and $1 \leq l \leq r$), the resume of C' and C'' is obtained as following (where $\cap$ denotes the intersection operator between intervals):

$$C' \oplus^{P} C'' = \langle x, y, \langle\langle [\max(c'_i, c''_i), \min(d'_i, d''_i)], p_i\rangle, i = 1, \dots, \min(k, l)\rangle\rangle\rangle$$

Example 4(k). Given the two PyP_BoDs $C' = \langle RT, NA2, \langle\langle [0,24], low\rangle, \langle [0,24], medium\rangle, \langle [0,24], high\rangle\rangle\rangle$ (the original constraint in the PyP_BoD_G between nodes RT and $NA2$) and $C'' = \langle RT, NA2, \langle\langle [10,39], low\rangle, \langle [11,37], medium\rangle, \langle [12,36], high\rangle\rangle\rangle$ (see below), the result of the resume operation between them is $C' \oplus^{P} C'' = \langle RT, NA2, \langle\langle [10,24], low\rangle, \langle [11,24], medium\rangle, \langle [12,24], high\rangle\rangle\rangle$.

Notably, in the constraint propagation algorithm, if an empty interval is obtained at a level *i* (i.e., the resume operator computes as intersection a degenerate interval), there is an inconsistency at level *i* and all the AdIs higher or equal than *i* can be removed from the set of PyP_BoDs. If an empty interval is obtained at the first level of the PyP_BoD, all the PyP_BoDs are inconsistent.

The extension operator $\odot^{P}$, at each level *i* of the pyramids, computes the new AdI by summing pairwise the starting points and the ending points of the input AdIs at level *i*. It pairs the resulting AdI with the preference corresponding with that level, until it reaches the highest level for both constraints.

Definition 29. Extension ($\odot^{P}$). Given a scale S_r, and given two PyP_BoDs $C' = \langle x, y, \langle\langle [c'_1, d'_1], p_1\rangle, \dots, \langle [c'_k, d'_k], p_k\rangle\rangle\rangle$ and $C'' = \langle y, z, \langle\langle [c''_1, d''_1], p_1\rangle, \dots, \langle [c''_l, d''_l], p_l\rangle\rangle\rangle$ (with $1 \leq k \leq r$ and $1 \leq l \leq r$), their extension is defined as following:

$$C' \odot^{P} C'' = \langle x, z, \langle\langle [c'_i + c''_i, d'_i + d''_i], p_i\rangle, i = 1, \dots, \min(k, l)\rangle\rangle$$

Example 4(l). Given the two PyP_BoDs $C' = \langle RT, NA1, \langle\langle [0,24], low\rangle, \langle [0,24], medium\rangle, \langle [0,24], high\rangle\rangle\rangle$ and $C'' = \langle NA1, NA2, \langle\langle [10,15], low\rangle, \langle [11,13], medium\rangle, \langle [12,12], high\rangle\rangle\rangle$, the result of the extension operation between them is $C' \odot^{P} C'' = \langle RT, NA2, \langle\langle [10,39], low\rangle, \langle [11,37], medium\rangle, \langle [12,36], high\rangle\rangle\rangle$.

Complexity. Because the operations of resume and extension perform, for each preference value, an intersection and a sum of two intervals and such basic operations can be computed in constant time, both the $\oplus^{\mathrm{P}}$ and $\odot^{\mathrm{P}}$ operators' time complexity is $O(r)$, i.e., they are linear in the number of preference values in S_r.

Notably, also the $\oplus^{\mathrm{P}}$ and $\odot^{\mathrm{P}}$ operators, as $\oplus^{\mathrm{PyPB}}$ and $\odot^{\mathrm{PyPB}}$, can deal both with discrete and dense domains.

To complete our definition of *closed semiring*, we now add our definition of the identity ***1*** for $\odot^{\mathrm{P}}$ and of the identity ***0*** for $\oplus^{\mathrm{P}}$. As an intuition, the identity ***0*** for $\oplus^{\mathrm{P}}$ corresponds to the non-existing constraint, and the identity ***1*** for $\odot^{\mathrm{P}}$ represents the distance between a point and itself.

Identities for $\odot^{P}$ and $\oplus^{P}$. Given a Scale S_r with cardinality r, the identity ***1*** for $\odot^{P}$ is $\bot = \langle\langle[0,0], p_1\rangle, \dots, \langle[0,0], p_r\rangle\rangle$ since, given any PyP_BoD $C = \langle\langle[c_1, d_1], p_1\rangle, \dots, \langle[c_r, d_r], p_r\rangle\rangle$[9] of the form $(\mathbb{A}^{\mathbb{D}} \times \mathbb{S})^r$, $C \odot^{P} \langle\langle[0,0], p_1\rangle, \dots, \langle[0,0], p_r\rangle\rangle = \langle\langle[0,0], p_1\rangle, \dots, \langle[0,0], p_r\rangle\rangle \odot^{P} C = C$.

The identity ***0*** for $\oplus^{P}$ is $\top = \langle\langle[-\infty, +\infty], p_1\rangle, \dots, \langle[-\infty, +\infty], p_r\rangle\rangle$ since, given any PyP_BoD $C = \langle\langle[c_1, d_1], p_1\rangle, \dots, \langle[c_r, d_r], p_r\rangle\rangle$ of the form $(\mathbb{A}^{\mathbb{D}} \times \mathbb{S})^r$, $C \oplus^{P} \langle\langle[-\infty, +\infty], p_1\rangle, \dots, \langle[-\infty, +\infty], p_r\rangle\rangle = \langle\langle[-\infty, +\infty], p_1\rangle, \dots, \langle[-\infty, +\infty], p_r\rangle\rangle \oplus^{P} C = C$.

$\langle\langle[-\infty, +\infty], p_1\rangle, \dots, \langle[-\infty, +\infty], p_r\rangle\rangle$ is also the annihilator for operator $\odot^{P}$ since, given any PyP_BoD $C = \langle\langle[c_1, d_1], p_1\rangle, \dots, \langle[c_r, d_r], p_r\rangle\rangle$ of the form $(\mathbb{A}^{\mathbb{D}} \times \mathbb{S})^r$,

$C \odot^{P} \langle\langle[-\infty, +\infty], p_1\rangle, \dots, \langle[-\infty, +\infty], p_r\rangle\rangle = \langle\langle[-\infty, +\infty], p_1\rangle, \dots, \langle[-\infty, +\infty], p_r\rangle\rangle \odot^{P} C = \langle\langle[-\infty, +\infty], p_1\rangle, \dots, \langle[-\infty, +\infty], p_r\rangle\rangle$.

We can finally show that, indeed, we have defined a closed semiring.

Property 4. $\langle \mathbb{PyP\_B}\circledcirc\mathbb{D}, \oplus^{P}, \odot^{P}, \bot, \top\rangle$ is a *closed semiring*.

Proof (sketch). The operators $\oplus^{P}$ and $\odot^{P}$ are both ***closed*** over $\mathbb{PyP\_B}\circledcirc\mathbb{D}$ by definition. $\oplus^{P}$ is ***associative***, because it performs two associative operators, i.e., maximum and minimum, to the endpoints of the AdIs. Analogously, $\oplus^{P}$ is also ***idempotent*** and ***commutative***. Therefore, as $\bot$ is the identity for $\oplus^{P}$, $\langle \mathbb{PyP\_B}\circledcirc\mathbb{D}, \oplus^{P}, \bot\rangle$ is a ***idempotent and commutative monoid***. $\odot^{P}$ is ***associative***, because it performs an associative operation, i.e., the pairwise sum, to the endpoints of the AdIs. Therefore, as $\top$ is the identity for $\odot^{P}$, $\langle \mathbb{PyP\_B}\circledcirc\mathbb{D}, \odot^{P}, \top\rangle$ is a ***monoid***. Finally, $\top$ is an **annihilator** for $\odot^{P}$ and, since pairwise sum distributes over the maximum and minimum operators, $\odot^{P}$ ***distributes*** over finite and countably infinite $\oplus^{P}$. Thus, $\langle \mathbb{PyP\_B}\circledcirc\mathbb{D}, \oplus^{P}, \odot^{P}, \bot, \top\rangle$ is a *closed semiring*. ■

Given the above proof, we can thus use our instantiation of Compute-Summaries to compute the minimal network of a graph of PyP_BoD constraints. Obviously, such a computation exploits the computational efficiency of Compute-Summaries:

Property 5. The complexity of the Compute-Summaries algorithm instantiated on the resume and extend operators $\oplus^{P}$ and $\odot^{P}$ is $\Theta(|V|^3 \cdot r)$.

Proof (sketch). In Compute-Summaries, $\oplus$ and $\odot$ are applied $|V|^3$ times. In our definition, they operate separately (in constant time) on each layer, and, given a scale S_r, layers are at most r.

Example 4(m). In Figure 5, we show the result of the application of Compute-Summaries (i.e., the minimal network) to the PyP_BoD_G of Figure 3. Due to readability reasons, in the figure we only show the most significative PyP_BoDs. In particular, with red color and bold text we represent the constraints modified or created by the application of Compute-Summaries. For instance, let us notice that the constraint between the time points RT and $NA2$, which originally was $\langle RT, NA2, \langle\langle[0,24], low\rangle, \langle[0,24], medium\rangle, \langle[0,24], high\rangle\rangle\rangle$, has been restricted to $\langle RT, NA2, \langle\langle[20,24], low\rangle, \langle[22,24], medium\rangle, \langle[23,24], high\rangle\rangle\rangle$ (thus restricting the span of time in which the second administration of nalidixic acid can be performed). On the other hand, in the original graph, there was no constraint between RT and $CC1$. Instead, in the minimal network, we have the constraint

[9] For the sake of convenience, in the rest of the paper we included in the constraints also the variables; here we indicate only the constraint itself.

$\langle RT, CC1, \langle\langle[6,10], low\rangle, \langle[6,9], medium\rangle, \langle[7,8], high\rangle\rangle\rangle$ (restricting the span of time in which the first administration of calcium carbonate can be performed).

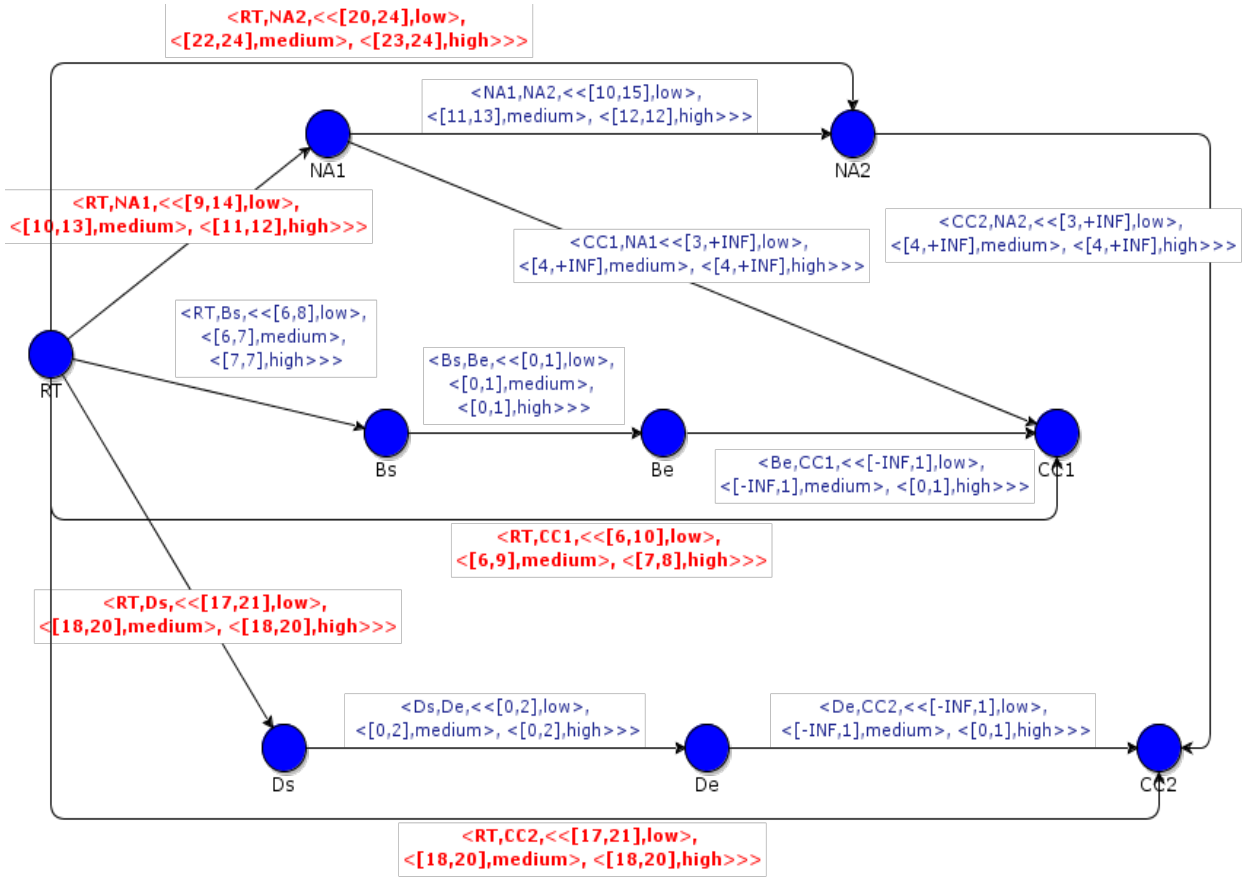

Figure 5. Minimal network of the PyP_BoD_G of Figure 3.

In this section, we have introduced a restricted form of aggregation operators for PyP_BoDs. Property 6 shows that the operators $\bigoplus^{P}$ and $\bigodot^{P}$ are an instance of the operators $\bigoplus^{PyPB}$ and $\bigodot^{PyPB}$ in case the min function is used as the $\bigoplus^{pref}$ operator and the max function is used as the $\bigodot^{pref}$ operator. In this way, we show that the formalism in this section is a particular case of the formalism in Section 4.

Property 6. Given a scale S_r, and given two PyP_BoDs $C' = \langle x, y, \langle\langle[c'_1, d'_1], p_1\rangle, \ldots, \langle[c'_k, d'_k], p_k\rangle\rangle\rangle$ and $C'' = \langle x, y, \langle\langle[c''_1, d''_1], p_1\rangle, \ldots, \langle[c''_l, d''_l], p_l\rangle\rangle\rangle$ (with $1 \leq k \leq r$ and $1 \leq l \leq r$), assuming that $\bigoplus^{pref}$ corresponds to the ***min*** function and that $\bigodot^{pref}$ corresponds to the ***max*** function, $C' \bigoplus^{PyPB} C'' = C' \bigoplus^{P} C''$ and $C' \bigodot^{PyPB} C'' = C' \bigodot^{P} C''$.

Proof (sketch). Let us consider the resume operators $\bigoplus^{PyPB}$ and $\bigoplus^{P}$. The proof for the extend operators $\bigodot^{PyPB}$ and $\bigodot^{P}$ is similar. We want to prove that $C' \bigoplus^{PyPB} C'' = C' \bigoplus^{P} C''$. Let us denote the result of $C' \bigoplus^{PyPB} C'' = C' \bigoplus^{P} C''$ as $\langle x, y, \langle\langle[c'''_1, d'''_1], p_1\rangle, \ldots, \langle[c'''_m, d'''_m], p_m\rangle\rangle\rangle$.

We prove the property considering two cases. The first case consists in proving that both operators compute the same top level $\langle[c'''_m, d'''_m], p_m\rangle$ of the result. This corresponds to proving that both $\bigoplus^{PyPB}$ and $\bigoplus^{P}$ determine that the values in $[c'''_m, d'''_m]$ are all and only those values which have a preference value of p_m. The second case consists in proving that for an arbitrary non-top level i, $1 \leq i < m$, both operators determine that $\langle[c'''_i, d'''_i], p_i\rangle$ is in the result, and that the values in $[c'''_i, d'''_i]$ are all and only those values which have preference value of p_i or higher.

First case. We prove that $\langle[c'''_m, d'''_m], p_m\rangle$ is the top level of $C' \oplus^{PyPB} C''$ if and only if $\langle[c'''_m, d'''_m], p_m\rangle$ is the top level of $C' \oplus^{P} C''$.

If $\langle[c'''_m, d'''_m], p_m\rangle$ is the top level of $C' \oplus^{P} C''$ then, for the semantics of Pyp_BoDs and by the definition of $\oplus^P$, there is no other level n, $m < n \leq r$, such that $[c'_n, d'_n] \cap [c''_n, d''_n] \neq \emptyset$ where $\langle[c'_n, d'_n], p_n\rangle \in C'$ and $\langle[c''_n, d''_n], p_n\rangle \in C''$ (otherwise $\langle[c'''_m, d'''_m], p_m\rangle$ would not be at the top level). Thus, there is no value in both the domains of C' and C'' which has a preference value higher than m both in C' and in C''. In other words, either in C' or in C'' every value in $[c'''_m, d'''_m]$ has m as maximum preference value. Thus, for the definition of $\oplus^{PyPB}$, $\langle[c'''_m, d'''_m], p_m\rangle$ is the top level of $C' \oplus^{PyPB} C''$.

If $\langle[c'''_m, d'''_m], p_m\rangle$ is the top level of $C' \oplus^{PyPB} C''$ then, for the semantics of Pyp_BoDs and by the definition of $\oplus^{PyPB}$, $[c'''_m, d'''_m]$ is an interval whose preference value both in C' and in C'' is at least p_m and there is no other interval for which there is a preference value higher than p_m both in C' and in C''. Thus, there is no other level n, $m < n \leq r$, such that $\langle[c'_n, d'_n], p_n\rangle \in C'$, $\langle[c''_n, d''_n], p_n\rangle \in C''$ and $[c'_n, d'_n] \cap [c''_n, d''_n] \neq \emptyset$, and, therefore, $\langle[c'''_m, d'''_m], p_m\rangle \in C' \oplus C''$.

Second case. We prove that $\langle[c'''_i, d'''_i], p_i\rangle$, $1 \leq i < m$, is a non-top level of $C' \oplus^{PyPB} C''$ if and only if $\langle[c'''_i, d'''_i], p_i\rangle \in C' \oplus^{P} C''$ at the same (non-top) level.

If $\langle[c'''_i, d'''_i], p_i\rangle \in C' \oplus^{PyPB} C''$, for the semantics of Pyp_BoDs, $[c'''_i, d'''_i]$ contains all and only the values that have a preference value of p_i or higher. By the definition of $\oplus^{PyPB}$, $[c'''_i, d'''_i]$ is built by performing the union of all the fragments returned by the $F^{\oplus}$ function that have a preference value of p_i or higher. For the definition of the $F^{\oplus}$ function (where the *min* function is used as $\oplus^{Pref}$), these fragments contain the values that both in C' and C'' have a preference value of at least p_i (otherwise the value of the minimum would be less than p_i). For the semantics of PyP_BoDs, the values of C' that have a preference value of at least p_i are all and only the values in $[c'_i, d'_i]$ such that $\langle[c'_i, d'_i], p_i\rangle \in C'$ and the values of C'' that have a preference value of at least p_i are all and only the values in $[c''_i, d''_i]$ such that $\langle[c''_i, d''_i], p_i\rangle \in C''$. Thus, the values that have a preference value of at least p_i in both PyP_BoDs are all and only in their intersection $[c'''_i, d'''_i] = [c'_i, d'_i] \cap [c''_i, d''_i]$, and $\langle[c'_i, d'_i] \cap [c''_i, d''_i], p_i\rangle \in C' \oplus^{P} C''$.

If $\langle[c'''_i, d'''_i], p_i\rangle \in C' \oplus^{P} C''$, for the semantics of PyP_BoDs, the values in $[c'''_i, d'''_i]$ are all and only the values both in C' and C'' that have a preference value of at least p_i. By the definition of $\oplus^P$, there exist two intervals $\langle[c'_i, d'_i], p_i\rangle \in C$ and $\langle[c''_i, d''_i], p_i\rangle \in C''$ such that $[c'''_i, d'''_i] = [c'_i, d'_i] \cap [c''_i, d''_i]$. By the semantics of PyP_BoDs, $[c'_i, d'_i]$ and $[c''_i, d''_i]$ are all and only the values with a preference value of at least p_i in C' and C'' respectively. By the definition of $\oplus^{PyPB}$, the operator performs the union of all the fragments returned by the $F^{\oplus}$ function that have a preference value of p_i or higher. For the definition of the $F^{\oplus}$ function (where the *min* function is used as $\oplus^{Pref}$), these fragments contain the values that both in C' and C'' have a preference value of at least p_i (otherwise the value of the minimum would be less than p_i), i.e., the values in $[c'_i, d'_i] \cap [c''_i, d''_i]$. Thus, $[c'''_i, d'''_i] \in C' \oplus^{PyPB} C''$. ■

As shown in the following corollary, the formalism introduced in this section is a particular case of the formalism in Section 3 since, given Properties 4 and 6 above, we have that the operators $\oplus^P$ and $\odot^P$ are also an instance of the operators $\oplus^{PB}$ and $\odot^{PB}$.

Corollary. Given two PyP_BoDs $C' = \langle x, y, \langle\langle[\mathrm{c'_1}, \mathrm{d'_1}], \mathrm{p_1}\rangle, \ldots, \langle[\mathrm{c'_k}, \mathrm{d'_k}], \mathrm{p_k}\rangle\rangle\rangle$ and $\mathrm{C''} = \langle x, y, \langle\langle[\mathrm{c''_1}, \mathrm{d''_1}], \mathrm{p_1}\rangle, \ldots, \langle[\mathrm{c''_l}, \mathrm{d''_l}], \mathrm{p_l}\rangle\rangle\rangle$,

$$\rho^{PyPB\rightarrow PB}(C' \oplus^{P} C'') = \rho^{PyPB\rightarrow PB}(C') \oplus^{\mathrm{PB}} \rho^{PyPB\rightarrow PB}(C'')$$

and, given two PyP_BoDs $C' = \langle x, y, \langle\langle [\mathrm{c'}_1, \mathrm{d'}_1], \mathrm{p}_1\rangle, \ldots, \langle [\mathrm{c'}_\mathrm{k}, \mathrm{d'}_\mathrm{k}], \mathrm{p}_\mathrm{k}\rangle\rangle\rangle$ and $\mathrm{C}'' = \langle y, z, \langle\langle [\mathrm{c}''_1, \mathrm{d}''_1], \mathrm{p}_1\rangle, \ldots, \langle [\mathrm{c}''_\mathrm{l}, \mathrm{d}''_\mathrm{l}], \mathrm{p}_\mathrm{l}\rangle\rangle\rangle$,

$$\rho^{PyPB\rightarrow PB}(C' \odot^{P} C'') = \rho^{PyPB\rightarrow PB}(C') \odot^{\mathrm{PB}} \rho^{PyPB\rightarrow PB}(C'').$$

Proof. Trivial given Properties 4 and 6. ■

6 QUERY ANSWERING FACILITIES

In this paper, we have proposed a family of approaches coping with BoDs with layered preferences. We have considered (i) the most general case, in which a preference can be associated with each distance, and user-defined operations are used to combine preferences (Section 3), (ii) the case in which preferences are "nested" (Section 4), and (iii) the case in which the *min* and *max* operations are used to combine "nested" preferences (Section 5). Though the complexity of the reasoning process in the three cases is different, in all cases the output is a minimal network representing the strictest distances between variables, and the preference of each distance. Intuitively speaking, such a minimal network represents the space of possible solutions, with their preferences. Such a result is very important: as we have already motivated in the introduction, in *decision support* systems, and in *mixed-initiative* approaches, the choice of a specific solution relies on users, within the space of possible ones. To support users in such a selection task, besides the minimal network, we also provide *query answering* facilities, to give users a way to explore the space of solutions. Such facilities are provided for approaches of all types (i)-(iii) mentioned above.

Notably, though in the following we exemplify the queries considering Example 4, in which PyP_BoDs are considered, the query facilities above can be applied (with different complexities in case propagation is needed) to all the formalisms discussed in this paper.

Now, first we give a formal definition in terms of the extended BNF grammar of our query language (see Figure 6), and second, we provide a number of different types of query that we support.

```
<Query>     ::= <HypQ> | <StandardQ>
<HypQ>      ::= <StandardQ> IF (Constr)+
<StandardQ> ::= {<BaseQList>} | PREFERENCE <Op> <Pref> | {<BoolQ>}
<BaseQList> ::= (<Var> ? <Var>)+
<BoolQ>     ::= <SBQ> | <CBQ>
```

Figure 6. Query language (extended BNF grammar).

Hypothetical queries (<HypQ>) are standard queries <StandardQ> that are answered by assuming that some additional constraints (indicated by $(\mathrm{Constr})^+$ in the extended grammar) hold.

We consider four types of standard queries.

BASIC EXTRACTION QUERIES (<BaseQList>) query the temporal distances (and their preferences) between pairs of variables in a list. Such queries are trivially answered by considering the constraints in the minimal network.

Example 4(o). In Example 4, the basic extraction query `(RT ? NA2)` asks for the temporal constraint between the reference time RT and the second administration of nalidixic acid NA2. The result of such a query for Example 4 is the constraint $\langle RT, NA2, \langle\langle[20,24], low\rangle, \langle[22,24], medium\rangle\langle[23,24], high\rangle\rangle\rangle$. ■

PREFERENCE QUERIES provide as output the constraints obtained by retaining (in each constraint) only those distances such that their preference p is in the relation <Op> with the query parameter <Pref>, where <Op> is a comparison operator (one of =, ≥, >), and <Pref> is a preference value in the given Scale. Then, the remaining constraints are propagated, using the proper reasoning algorithm, and the output constraints are provided as output.

Example 4(p). In Example 4, one might want to obtain only the constraints with preference greater or equal to *medium*. The corresponding query is `PREFERENCE ≥ medium`. Part of the result of such a query is shown in Figure 7 (such as for Figure 5, we show only the most interesting constraints). ■

BOOLEAN (<BoolQ>) QUERIES can be simple (SBQ) or composite (CBQ). Such queries ask whether one (SBQ) or more (CBQ) constraints between pairs of variables hold and return a Boolean value.

An SBQ v_i<(d_1 Op p)>v_j (e.g., "v_1<(3 ≥ p>v_2") is answered by considering whether the distance between v_i and v_j can be d_1 in the corresponding constraint in the minimal network, and whether its preference satisfies the condition imposed by the Op operator.

Example 4(q). In Example 4, one may want to know whether the second administration of Nalidixic Acid can be administered at 22, with preference *high*, or not. The answer to query `RT(22 = high)NA2`, in Example 4, is `FALSE`.

A CBQ query is a set (conjunction) of SBQ queries. Answering a CBQ query (e.g., "{v_1<(3 ≥ p)>v_2, v_1<(5 ≥ p')>v_3}) requires four steps:

(1) For each SBQ v_i <(d_1 Op p)> v_j all the distances except d_1 are removed from the corresponding constraint relating v_i and v_j in the minimal network.

(2) (An instantiation of) the proper constraint-propagation algorithm (i.e., path consistency with resume and extend in Definitions 9 and 10, or Compute-Summaries with resume and extend in Definitions 22 and 25, or Compute-Summaries with resume and extend in Definitions 28 and 29) propagates the resulting constraints.

(3) If the resulting set of constraints is inconsistent, the answer is NO.

(4) Otherwise, each SBQ is checked separately. If at least one constraint is not satisfied, the answer is NO, otherwise the answer is YES.

Example 4(r). In Example 4, one may want to know whether Nalidixic Acid can be administered at 12 (first administration NA1) and at 24 (second administration NA2), both with preference *high*. The answer to query `{RT(12 = high)NA1, RT(24 = high)NA2}`, in Example 4, is `TRUE`.

Hypothetical queries (<HypQ>) are queries <StandardQ> as above, to be answered after assuming some additional constraints. To answer such queries, we first provisionally add, to the set of constraints, the new hypothetical constraints, and, then, we apply the proper constraint-propagation algorithm for obtaining the new tightest constraints. In case the new constraints are inconsistent with regard to the previous ones, a warning is given. Finally, <StandardQ>

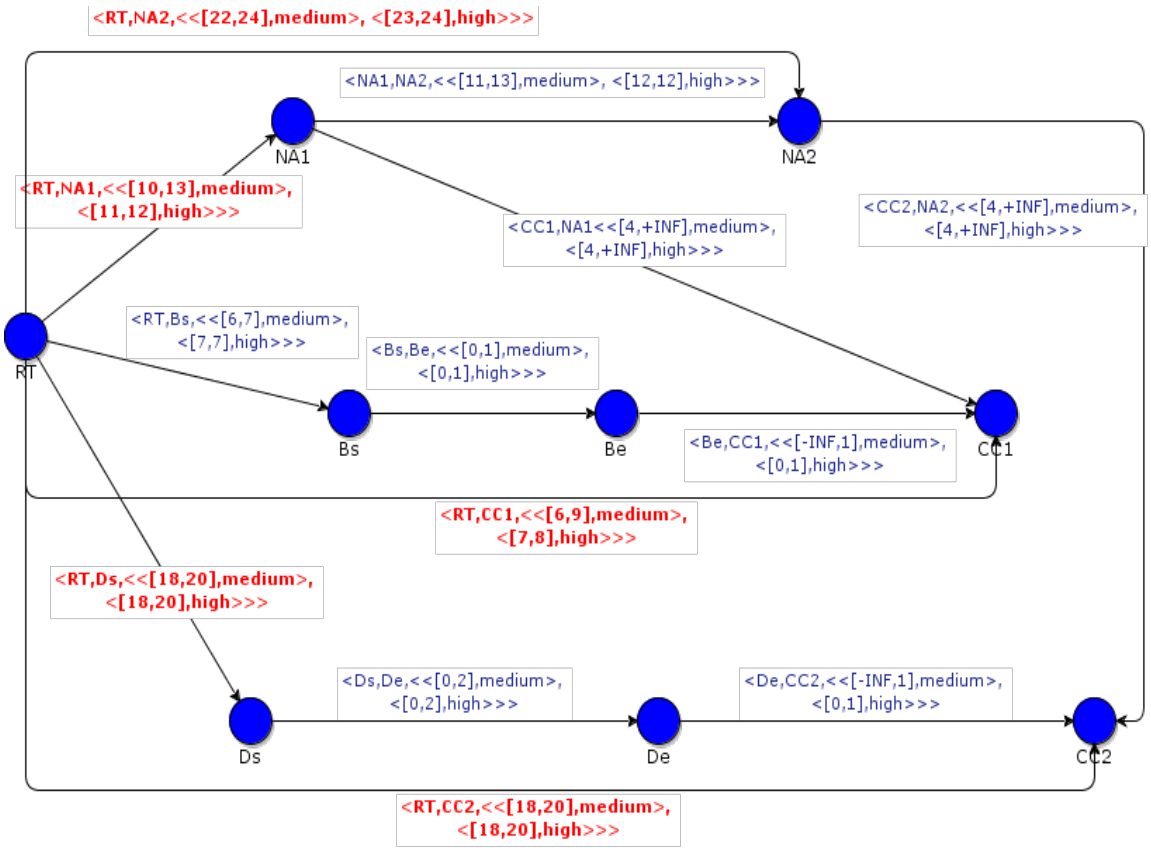

Figure 7. Result of the query "IP ≥ medium". For the sake of simplicity, the figure reports only the constraints shown in Figure 5.

is answered (as detailed above) on the basis of the new set of constraints. Notice also that, in order to add the new constraints, the resume operator must be used.

Example 4(s). For instance, considering Example 4, a user may want to know when she can take the second administration of nalidixic acid (NA2), if the first administration of nalidixic acid (NA1) is done at 10. The respective hypothetical query is

$$(NA2\ ?\ RT)IF\{\langle RT, NA1, \langle\langle[10{,}10], low\rangle, \langle[10{,}10], medium\rangle, \langle[10{,}10], high\rangle\rangle\rangle\}$$

The answer is the PyP_BoD constraint $\langle RT, NA2, \langle\langle[20{,}24], low\rangle, \langle[22{,}23], medium\rangle, \langle[22{,}22], high\rangle\rangle\rangle$. ■

7 Comparisons and Conclusions

The literature about CSP is moving from treating crisp constraint to dealing with fuzzy or probabilistic constraints, so that preferences and/or uncertainty can be supported, and a vast number of approaches have been proposed. Our work provides a significant number of original contributions along two main directions.

First, while in the context of crisp BoD constraints the problem of determining the "space of solutions" by means of the minimal network and of exploring it through *queries* have been widely addressed, the approaches considering preferences have focused their attention only on the problem of determining the optimal (i.e., the most preferred) solutions. Our approach overcomes such a main limitation (see points (iv) and (v) below), expanding the areas of applicability of non-crisp CSP techniques (e.g., to tasks such as decision making, or to mixed-initiative approaches).

Second, given the variety of solutions in the CSP area, to increase the generality and the applicability of our proposal, we do not propose one specific approach, but a whole family of approaches, parametric with respect to:

(i) the scale for layered preferences, and

(ii) the operations (*extend* and *resume*) to combine preferences during constraint propagation.

Notably, thanks to feature (ii) above, as far as we know, our approach is the first one in the context of constraints with preferences able to consider *user-defined* operations to combine preferences. Additionally,

(iii) we consider not only general preferences (Section 3), but also nested ones (Sections 4 and 5).

In particular, the identification and management of nested preferences are another original feature of our approach, leading to a user-friendly and compact representation, and to computational advantages.

For each family of approaches, we provide:

(iv) reasoning algorithms to compute the minimal network of constraints (with their preferences) covering the different cases;

(v) query answering facilities for exploring the minimal network covering the different cases.

As discussed above, issues (iv) and (v) have been widely addressed in the area of crisp BoDs, but not yet in the context of BoDs with preferences (except in [60], see the comparisons below).

Additionally, to address efficiency issues,

(vi) we exploit the property of *closed semirings* to efficiently compute the minimal network in case of nested preferences, and in case *min* and *max* are used to combine preferences.

No other approach in the literature covers such a wide range of phenomena within a single and homogeneous framework.

While a general overview of the mostly related works has been already presented in Section 2, here we conclude by comparing our approach with the two approaches that are more strictly related to ours: the one by Khatib et al. [37] (and its evolution [36]) and the one by Andolina et al. [60] .

Khatib et al. [37] take into account STP [19] (which is, intuitively, the temporal interpretation of BoDs; notably, also TCSP [19] – a form of disjunctions of BoDs— are also considered). Their approach is general and takes into account two main types of preference functions from values in the admissibility range of BoDs to the domain of preferences: (i) general functions and (ii) semi-convex functions. As regards the *extend* and *revise* operations to combine preferences, they focus their attention on operations forming a closed semiring (as we do in Section 5), providing as an example the semiring obtained by adopting the ***min*** function to ***resume*** preferences and ***max*** to extend them (that we also adopt in Section 5). The goal of constraint propagation in their approach is to determine the optimal solutions, i.e., solutions with the maximal preference. They explore the complexity of such a constraint propagation both in cases (i) and (ii) above, and provide an algorithm to find the optimal solutions in case (ii) studying the conditions under which it operates in polynomial time.

The similarities and differences between our approach and Khatib et al.'s one are schematized in Table 1, considering five main parameters: (i) (type of) constraints, (ii) preference functions, (iii) preference combination operations, (iv) goals of the constraint propagation algorithms, and (v) query answering facilities.

	Khatib et al.'s approach	This approach
Constraints	BoDs with temporal interpretation	BoDs (any interpretation)
	A form of disjunctions of BoDs with temporal interpretation	
Preference functions	Points→Any	Points→Scale
	Semiconvex	Nested preferences
Preference combination	Closed semiring	Any
		Closed semiring
Reasoning task	Optimal solutions only	Minimal network with preferences
Query answering	No	Yes

Table 1. Comparison between Khatib et al's approach [37] and the approach in this paper.

(i) **Constraints**. Both approaches operate on BoD constraints (though Khatib et al. not only consider STP –i.e., the temporal interpretation of BODs– but also take into account TCSP).
(ii) **Preference functions**. Both approaches consider both (i) general functions to associate preferences with BODs, and (ii) functions with specific features (functions leading to nested preferences in our approach, *semi-convex* functions in Khatib et al.'s approach). Notably, we focus on functions whose co-domain is a scale (i.e., a finite and ordered set of values), and, in such a context, the convex functions exactly correspond to our nested preference functions.
(iii) **Preference combination operators (extend and revise)**. Both approaches consider operations to combine preferences during constraint propagation that form a closed-semiring algebraic structure. However, we also generalize our approach to consider also *user-defined* resume and extend operations to combine preferences.
(iv) **Reasoning task**. The two approaches substantially differ about the tasks of constraint propagation. In Khatib et al., reasoning algorithms aim at identifying the optimal solutions (i.e., solutions with highest preference). On the other hand, our algorithms aim to determine a compact representation of the space of solutions with their preferences.
(v) **Query Answering**. Since we provide the space of solutions, we also provide query answering mechanisms to explore it, while this task is neglected in Khatib et al., where just optimal solutions are considered.

Notably, considering also user-defined preference combination operations (see point (iii)) greatly enhances the generality of our approach, and issues (iv) and (v) make our approach suitable also to tasks such as decision support and mixed-initiative approaches, in which the "exploration" and the choice between possible solutions must be left to the users. Indeed, as far as we know, the only approach in the literature coping with BoDs with preferences, and facing the problem of providing the minimal network instead of the optimal solutions is the one in Andolina et al. [60]. Andolina et al. have recently proposed an approach to fuzzy *temporal* constraints in which a numeric preference is associated with each possible distance between time points [60] (considering only discrete domains for distances). They have chosen a specific way to combine preferences, in such a way that the extend and resume operators form a *closed semiring*, and the Compute-Summaries algorithm can be exploited to evaluate the minimal network (as we do in the version of our approach concerning "nested" preferences; see Section 5). Additionally, it is the first approach in

the fuzzy-constraint literature that proposes query-answering facilities for exploring the minimal network with preferences. Our approach extends and generalizes [60] in many ways, namely:

(i) we operate on BoD constraints, with no specific commitment to their temporal interpretation,

(ii) we generalize the approach in [60] for operating also on continuous domains, while [60] supports discrete domains only,

(iii) we support user-defined layered preferences, while in [60] only numeric preferences are supported, and we take into account also nested distributions of preferences.

(iv) we support user-defined operators to combine preferences, while just one specific definition of *resume* and *extend* is supported by [33].

Notably, in many practical tasks and domains (e.g., in medicine) the treatment of nested preferences facilitates users, that are not restricted to assign to each distance a definite numeric preference value. In addition, in Sections 4 and 5 of this paper we move from general preferences to nested ones, which are not considered in [60]. Nested preferences are not only useful in many practical tasks and domains, but their adoption also involves several practical and theoretical advantages. They reduce both space and time complexity. Regarding space complexity, while in [60] for each constraint all possible distance values must be explicitly stored, together with their preference, our approach allows a more compact representation. Moreover, nested preferences reduce the time complexity of the algorithm computing the minimal network (which in [60] is $\theta(n^3 \cdot |D'|^2)$, where n is the number of points and D' is the domain of the distance values).

Moreover, Andolina et al. only consider a specific definition for the extend and revise operators to combine preferences, while Kathib et al. take into account only operations forming a closed semiring. Since many more options have been proposed in literature (see Section 2.2), and the choice between them seems domain-/task-dependent, we regard the fact that our general approach (see Sections 3 and 4) can support also user-defined operators to combine preferences (i.e., it is parametric with respect to such operators) as a relevant step forward in the state of the art, improving the applicability of constraint-based approaches.

Finally, the approach in this paper is a substantial extension of the one we have recently proposed in [4]. Indeed, such a work addresses the problems of determining the minimal network and querying it considering BoD constraints with preferences, but only in the context of "pyramid" constraints, and considering the *min* function to *resume* preferences and *max* to extend them (i.e., the approach in [4] constitutes the preliminary version of the "specific" approach presented in Section 5 of this paper).

As a final note, we would like to highlight that, although the approach we propose is totally domain- and task-independent, we devise to apply it within GLARE [14], a long-term project started in 1997 with one of the major hospitals in Italy, aiming at assisting physicians in the management of clinical guidelines with decision support techniques.

REFERENCES

1. Aho, A.V., Hopcroft, J.E.: The Design and Analysis of Computer Algorithms. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1974).
2. Allen, J.F.: Maintaining knowledge about temporal intervals. Communications of the ACM. 26, 11, 832–843 (1983). https://doi.org/10.1145/182.358434.
3. Anselma, L. et al.: An artificial intelligence framework for compensating transgressions and its application to diet management. Journal of biomedical informatics. 68, 58–70 (2017).

4. Anselma, L. et al.: Temporal Reasoning with Layered Preferences. In: International Symposium on Methodologies for Intelligent Systems. pp. 367–376 Springer (2018).
5. Anselma, L. et al.: Towards a comprehensive treatment of repetitions, periodicity and temporal constraints in clinical guidelines. Artificial Intelligence in Medicine. 38, 2, 171–195 (2006). https://doi.org/10.1016/j.artmed.2006.03.007.
6. Badaloni, S., Giacomin, M.: The algebra IAfuz: a framework for qualitative fuzzy temporal reasoning. Artificial Intelligence. 170, 10, 872–908 (2006).
7. Bard, J.F., Purnomo, H.W.: Hospital-wide reactive scheduling of nurses with preference considerations. IIE Transactions. 37, 7, 589–608 (2005). https://doi.org/10.1080/07408170590948468.
8. Barro, S. et al.: A model and a language for the fuzzy representation and handling of time. Fuzzy Sets and Systems. 61, 2, 153–175 (1994).
9. Bessière, C. et al.: An optimal coarse-grained arc consistency algorithm. Artificial Intelligence. 165, 2, 165–185 (2005).
10. Billiet, C. et al.: A comparison technique for ill-known time intervals. In: Fuzzy Systems (FUZZ-IEEE), 2016 IEEE International Conference on. pp. 1963–1969 IEEE (2016).
11. Bistarelli, S. et al.: From soft constraints to bipolar preferences: modelling framework and solving issues. Journal of Experimental & Theoretical Artificial Intelligence. 22, 2, 135–158 (2010).
12. Bistarelli, S. et al.: Labeling and partial local consistency for soft constraint programming. In: International Symposium on Practical Aspects of Declarative Languages. pp. 230–248 Springer (2000).
13. Bistarelli, S. et al.: Semiring-based constraint satisfaction and optimization. J. ACM. 44, 2, 201–236 (1997). https://doi.org/10.1145/256303.256306.
14. Bottrighi, A., Terenziani, P.: META-GLARE: A meta-system for defining your own computer interpretable guideline system—Architecture and acquisition. Artificial Intelligence in Medicine. 72, 22–41 (2016). https://doi.org/10.1016/j.artmed.2016.07.002.
15. Boutilier, C. et al.: CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements. J. Artif. Int. Res. 21, 1, 135–191 (2004).
16. Brusoni, V. et al.: On the Computational Complexity of Querying Bounds on Differences Constraints. Artif. Intell. 74, 2, 367–379 (1995).
17. Bugarín, A. et al.: Fuzzy Knowledge Representation for Linguistic Description of Time Series. In: IFSA-EUSFLAT. Atlantis Press (2015).
18. Cormen, T.T. et al.: Introduction to Algorithms. MIT Press, Cambridge, MA, USA (1990).
19. Dechter, R. et al.: Temporal Constraint Networks. Artificial Intelligence. 49, 1–3, 61–95 (1991). https://doi.org/10.1016/0004-3702(91)90006-6.
20. Dechter, R., Pearl, J.: Network-based heuristics for constraint-satisfaction problems. Artificial intelligence. 34, 1, 1–38 (1987).
21. Dubois, D. et al.: Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty. Appl Intell. 6, 4, 287–309 (1996). https://doi.org/10.1007/BF00132735.
22. Dubois, D. et al.: The calculus of fuzzy restrictions as a basis for flexible constraint satisfaction. In: [Proceedings 1993] Second IEEE International Conference on Fuzzy Systems. pp. 1131–1136 vol.2 (1993). https://doi.org/10.1109/FUZZY.1993.327356.
23. Dubois, D., Fargier, H.: On the qualitative comparison of sets of positive and negative affects. In: European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty. pp. 305–316 Springer (2005).
24. Dubois, D., Fargier, H.: Qualitative Decision Making with Bipolar Information. In: Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR-06). pp. 175–186 AAAI Press, Menlo Park, CA (2006).
25. Effinger, R.T. et al.: Dynamic controllability of temporally-flexible reactive programs. In: ICAPS. pp. 122–129 (2009).
26. Fang, C. et al.: Chance-Constrained Probabilistic Simple Temporal Problems. In: AAAI. pp. 2264–2270 (2014).
27. Fargier, H., Lang, J.: Uncertainty in constraint satisfaction problems: A probabilistic approach. In: Clarke, M. et al. (eds.) Symbolic and Quantitative Approaches to Reasoning and Uncertainty. pp. 97–104 Springer Berlin Heidelberg (1993).
28. Floyd, R.W.: Algorithm 97: Shortest Path. Commun. ACM. 5, 6, 345– (1962). https://doi.org/10.1145/367766.368168.
29. Fodor, J.C., Roubens, M.R.: Fuzzy Preference Modelling and Multicriteria Decision Support. Springer Netherlands (1994).
30. Freuder, E.C., Wallace, R.J.: Partial constraint satisfaction. Artificial Intelligence. 58, 1, 21–70 (1992). https://doi.org/10.1016/0004-3702(92)90004-H.
31. Gammoudi, A. et al.: Modeling temporal relations between Fuzzy TIME intervals: A disjunctive view. In: Fuzzy Systems (FUZZ-IEEE), 2016 IEEE International Conference on. pp. 50–57 IEEE (2016).

32. Grabisch, M. et al.: Fuzzy Aggregation of Numerical Preferences. In: Słowiński, R. (ed.) Fuzzy Sets in Decision Analysis, Operations Research and Statistics. pp. 31–68 Springer US, Boston, MA (1998). https://doi.org/10.1007/978-1-4615-5645-9_2.
33. Hunsberger, L. et al.: The Dynamic Controllability of Conditional STNs with Uncertainty. In: PlanEx 2012. pp. 121–128 (2012).
34. Jobczyk, K.A., Ligęza, A.: Towards a new convolution-based approach to the specification of STPU-solutions. In: Fuzzy Systems (FUZZ-IEEE), 2016 IEEE International Conference on. pp. 782–789 IEEE (2016).
35. Kamide, N., Koizumi, D.: Method for Combining Paraconsistency and Probability in Temporal Reasoning. Journal of advanced computational intelligence and intelligent informatics. 20, 5, 813–827 (2016).
36. Khatib, L. et al.: Solving and learning a tractable class of soft temporal constraints: Theoretical and experimental results. AI Communications. 20, 3, 181–209 (2007).
37. Khatib, L. et al.: Temporal constraint reasoning with preferences. In: Proceedings of the 17th international joint conference on Artificial intelligence-Volume 1. pp. 322–327 Morgan Kaufmann (2001).
38. Klement, E.P. et al.: Triangular Norms. Springer Netherlands (2000).
39. Li, M. et al.: Handling temporal constraints with preferences in HTN planning for emergency decision-making. Journal of Intelligent & Fuzzy Systems. 30, 4, 1881–1891 (2016).
40. Long, Z. et al.: Efficient path consistency algorithm for large qualitative constraint networks. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence. pp. 1202–1208 AAAI Press (2016).
41. Mackworth, A.: Constraint Satisfaction. In: Encyclopedia of AI. pp. 205–211 Springer Verlag (1988).
42. Mizumoto, M.: Pictorial representations of fuzzy connectives, Part I: Cases of t-norms, t-conorms and averaging operators. Fuzzy Sets and Systems. 31, 2, 217–242 (1989). https://doi.org/10.1016/0165-0114(89)90005-5.
43. Moffitt, M.D.: On the modelling and optimization of preferences in constraint-based temporal reasoning. Artificial Intelligence. 175, 7, 1390–1409 (2011). https://doi.org/10.1016/j.artint.2010.11.016.
44. Montanari, U.: Networks of constraints: Fundamental properties and applications to picture processing. Information Sciences. 7, 95–132 (1974). https://doi.org/10.1016/0020-0255(74)90008-5.
45. Mouhoub, M., Liu, J.: Managing uncertain temporal relations using a probabilistic Interval Algebra. In: 2008 IEEE International Conference on Systems, Man and Cybernetics. pp. 3399–3404 (2008). https://doi.org/10.1109/ICSMC.2008.4811823.
46. Mouhoub, M., Sukpan, A.: Managing temporal constraints with preferences. Spatial Cognition & Computation. 8, 1–2, 131–149 (2008).
47. Müller-Hannemann, M. et al.: Timetable Information: Models and Algorithms. In: Geraets, F. et al. (eds.) Algorithmic Methods for Railway Optimization, International Dagstuhl Workshop, Dagstuhl Castle, Germany, June 20-25, 2004, 4th International Workshop, ATMOS 2004, Bergen, Norway, September 16-17, 2004, Revised Selected Papers. pp. 67–90 Springer (2004). https://doi.org/10.1007/978-3-540-74247-0_3.
48. Planken, L.R. et al.: Computing all-pairs shortest paths by leveraging low treewidth. Journal of Artificial Intelligence Research. 43, 353–388 (2012).
49. Rosenfeld, A. et al.: Scene Labeling by Relaxation Operations. IEEE Transactions on Systems, Man, and Cybernetics. SMC-6, 6, 420–433 (1976). https://doi.org/10.1109/TSMC.1976.4309519.
50. Rossi, F. et al.: Preferences in Constraint Satisfaction and Optimization. 1. 29, 4, 58–58 (2008). https://doi.org/10.1609/aimag.v29i4.2202.
51. Ruttkay, Z.: Fuzzy constraint satisfaction. In: Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference. pp. 1263–1268 vol.2 (1994). https://doi.org/10.1109/FUZZY.1994.343640.
52. Ryabov, V., Trudel, A.: Probabilistic temporal interval networks. In: Proceedings. 11th International Symposium on Temporal Representation and Reasoning, 2004. TIME 2004. pp. 64–67 (2004). https://doi.org/10.1109/TIME.2004.1314421.
53. Santana, P., Williams, B.C.: Chance-constrained consistency for probabilistic temporal plan networks. 24th ICAPS. 272–279 (2014).
54. Schiex, T.: Possibilistic Constraint Satisfaction Problems or “How to Handle Soft Constraints?” In: Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence. pp. 268–275 Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1992).
55. Schiex, T. et al.: Valued Constraint Satisfaction Problems: Hard and Easy Problems. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1. pp. 631–637 Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1995).
56. Schwalb, E., Vila, L.: Temporal Constraints: A Survey. Constraints. 3, 2–3, 129–149 (1998). https://doi.org/10.1023/A:1009717525330.
57. Shapiro, L.G., Haralick, R.M.: Structural Descriptions and Inexact Matching. IEEE Transactions on Pattern Analysis and Machine Intelligence. PAMI-3, 5, 504–519 (1981). https://doi.org/10.1109/TPAMI.1981.4767144.
58. Shimbel, A.: Structural parameters of communication networks. Bulletin of Mathematical Biophysics. 15, 4, 501–507 (1953). https://doi.org/10.1007/BF02476438.

59. Taş, D. et al.: The time-dependent vehicle routing problem with soft time windows and stochastic travel times. Transportation Research Part C: Emerging Technologies. 48, 66–83 (2014). https://doi.org/10.1016/j.trc.2014.08.007.
60. Terenziani, P. et al.: Managing Temporal Constraints with Preferences: Representation, Reasoning, and Querying. IEEE Trans. Knowl. Data Eng. 29, 9, 2067–2071 (2017). https://doi.org/10.1109/TKDE.2017.2697852.
61. Terenziani, P.: Reasoning about Time. In: Encyclopedia of Cognitive Science. pp. 869–874 John Wiley & Sons, Ltd (2006).
62. Terenziani, P., Andolina, A.: Probabilistic quantitative temporal reasoning. In: Proceedings of the Symposium on Applied Computing. pp. 965–970 ACM (2017).
63. Torchio, P.T. and G.M. and M.: A modular approach for representing and executing clinical guidelines. Artificial Intelligence in Medicine. 23, 3, 249–276 (2001).
64. Torrens, M., Faltings, B.: Using Soft CSPs for Approximating Pareto-Optimal Solution Sets. In: In AAAI Workshop Proceedings Preferences in AI and CP: Symbolic Approaches. AAAI Press (2002).
65. Tsamardinos, I.: A probabilistic approach to robust execution of temporal plans with uncertainty. In: Hellenic Conference on Artificial Intelligence. pp. 97–108 Springer (2002).
66. Van Beek, P., Cohen, R.: Approximation algorithms for temporal reasoning. University of Waterloo. Department of Computer Science (1989).
67. Venable, K.B. et al.: Weak and dynamic controllability of temporal problems with disjunctions and uncertainty. In: Workshop on constraint satisfaction techniques for planning & scheduling. pp. 50–59 (2010).
68. Vidal, T.: Handling contingency in temporal constraint networks: from consistency to controllabilities. Journal of Experimental & Theoretical Artificial Intelligence. 11, 1, 23–45 (1999).
69. Vila, L.: A Survey on Temporal Reasoning in Artificial Intelligence. AI Commun. 7, 1, 4–28 (1994).
70. Vila, L., Godo Lacasa, L.: On fuzzy temporal constraint networks. Mathware & soft computing. 1, 3, 315–334 (1994).
71. Vilain, M.B., Kautz, H.A.: Constraint Propagation Algorithms for Temporal Reasoning. In: Kehler, T. (ed.) Proceedings of the 5th National Conference on Artificial Intelligence. Philadelphia, PA, August 11-15, 1986. Volume 1: Science. pp. 377–382 Morgan Kaufmann (1986).
72. Wang, H.-F., Wu, K.-Y.: Preference Approach to Fuzzy Linear Inequalities and Optimizations. Fuzzy Optim Decis Making. 4, 1, 7–23 (2005). https://doi.org/10.1007/s10700-004-5567-0.
73. Xu, L., Choueiry, B.: A new effcient algorithm for solving the simple temporal problem. In: Temporal Representation and Reasoning, 2003 and Fourth International Conference on Temporal Logic. Proceedings. 10th International Symposium on. pp. 210–220 IEEE (2003).
74. Yang, R. et al.: Some realizations and instances of Yager prioritized preference frame with application in evaluation and decision making. (2020). https://doi.org/10.1002/int.22218.
75. Yorke-Smith, N. et al.: Temporal reasoning with preferences and uncertainty. In: IJCAI. pp. 1385–1386 (2003).

Appendix.

An algorithm describing the fragments function in Definition 20.

fragments([s,e], {[s_1,e_1], …, [s_k,e_k]}) {

P ← $\{a_i \mid a_i \in \{s, e, s_1, e_1, \ldots, s_k, e_k\} \wedge s \leq a_i \leq e\}$

R ← {}

Card ←|P|-1

for i←1…Card **do**

c ← min(P)

P ← P – {c}

d ← min(P)

R ← R ∪ {[c,d]}

return R

}